

System Design and Methodology / Embedded Systems Design

X. System-Level Power/Energy Optimization

**TDTS07/TDDI08
VT 2026**

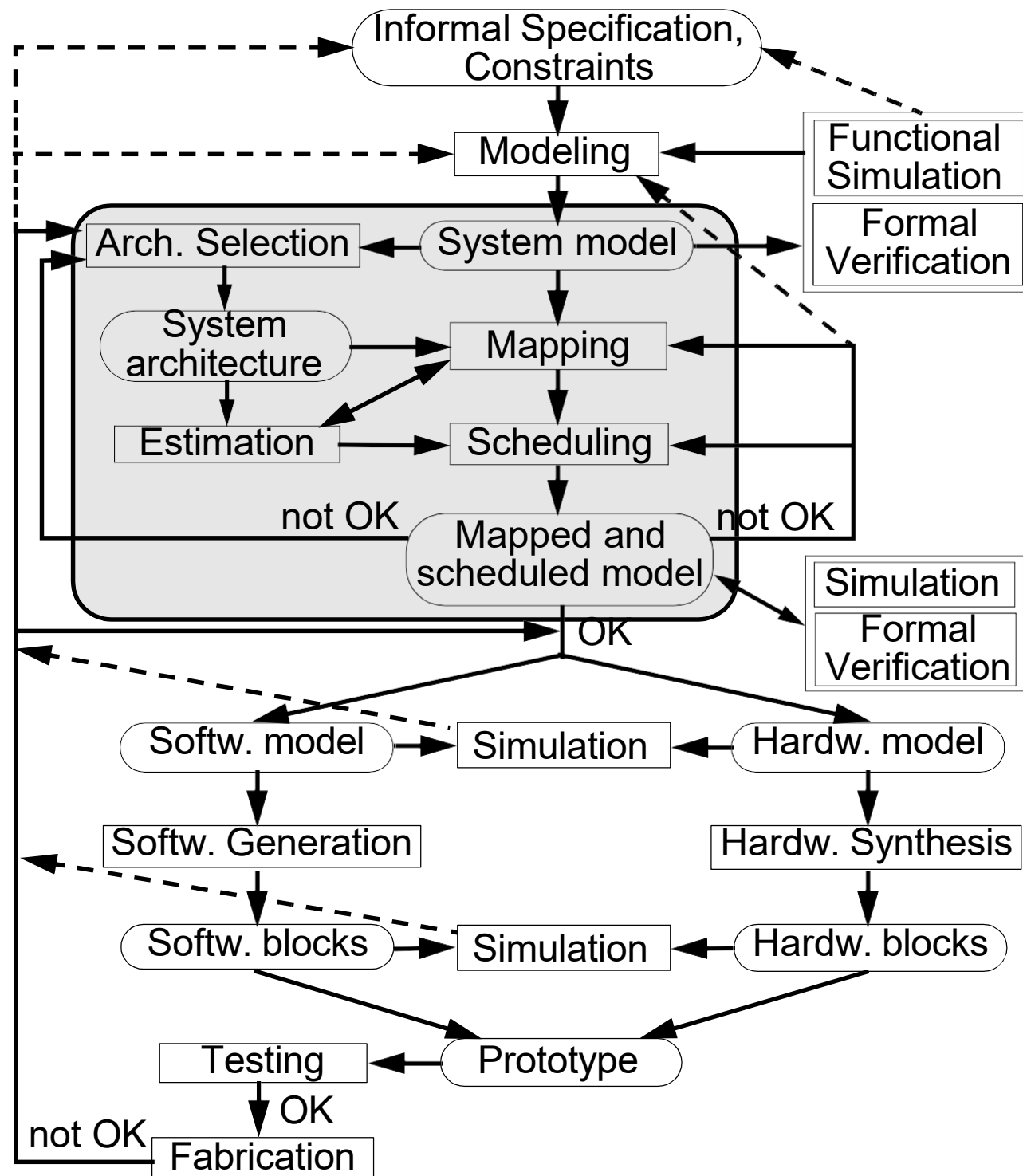
Ahmed Rezine

(Based on material by Petru Eles and Soheil Samii)

**Institutionen för datavetenskap (IDA)
Linköpings universitet**

SYSTEM-LEVEL POWER/ENERGY OPTIMIZATION

1. Sources of Power Dissipation
2. Reducing Power Consumption
3. System Level Power Optimization
4. Dynamic Power Management
5. Mapping and Scheduling for Low Energy
6. Real-Time Scheduling with Dynamic Voltage Scaling



Why is Power Consumption an Issue?

- Portable systems: battery life time!
- Systems with limited power budget: Mars Pathfinder, autonomous helicopter, ...
- Desktops and servers: high power consumption
 - raises temperature and deteriorates performance & reliability
 - increases the need for expensive cooling mechanisms
- One main difficulty with developing high performance chips is heat extraction.
- High power consumption has economical and ecological consequences.

Sources of Power Dissipation in CMOS Devices

$$P = \frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW} + Q_{SC} \times V_{DD} \times f \times N_{SW} + I_{leak} \times V_{DD}$$

C = node capacitances

N_{SW} = switching activities
(number of gate transitions per clock cycle)

f = frequency of operation

V_{DD} = supply voltage

Q_{SC} = charge carried by short circuit current per transition

I_{leak} = leakage current

Sources of Power Dissipation in CMOS Devices

$$P = \underbrace{\frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}}_{\text{dynamic}} + Q_{SC} \times V_{DD} \times f \times N_{SW} + I_{leak} \times V_{DD}$$

C = node capacitances

N_{SW} = switching activities
(number of gate transitions per clock cycle)

f = frequency of operation

V_{DD} = supply voltage

Q_{SC} = charge carried by short circuit current per transition

I_{leak} = leakage current

Sources of Power Dissipation in CMOS Devices

$$P = \underbrace{\frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}}_{\text{Switching power}} + \underbrace{Q_{SC} \times V_{DD} \times f \times N_{SW}}_{\text{Short-circ. power}} + I_{leak} \times V_{DD}$$

Switching power
Power required to charge/discharge circuit nodes

Short-circ. power
Dissipation due to short-circuit current

C = node capacitances

N_{SW} = switching activities
(number of gate transitions per clock cycle)

f = frequency of operation

V_{DD} = supply voltage

Q_{SC} = charge carried by short circuit current per transition

I_{leak} = leakage current

Sources of Power Dissipation in CMOS Devices

$$P = \underbrace{\frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}}_{\text{Switching power}} + \underbrace{Q_{SC} \times V_{DD} \times f \times N_{SW}}_{\text{Short-circ. power}} + \underbrace{I_{leak} \times V_{DD}}_{\text{Leakage power}}$$

Switching power
Power required to charge/discharge circuit nodes

Short-circ. power
Dissipation due to short-circuit current

Leakage power
Dissipation due to leakage current

C = node capacitances

N_{SW} = switching activities
(number of gate transitions per clock cycle)

f = frequency of operation

V_{DD} = supply voltage

Q_{SC} = charge carried by short circuit current per transition

I_{leak} = leakage current

Sources of Power Dissipation in CMOS Devices

$$P = \underbrace{\frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}}_{\text{Switching power}} + \underbrace{Q_{SC} \times V_{DD} \times f \times N_{SW}}_{\text{Short-circ. power}} + \underbrace{I_{leak} \times V_{DD}}_{\text{Leakage power}}$$

Switching power
Power required to charge/discharge circuit nodes

Short-circ. power
Dissipation due to short-circuit current

Leakage power
Dissipation due to leakage current

- Earlier:
Leakage power has been considered negligible compared to dynamic.
- Today:
Total dissipation from leakage is approaching the total from dynamic.
- As transistor sizes shrink:
Leakage power becomes significant.

Sources of Power Dissipation in CMOS Devices

$$P = \underbrace{\frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}}_{\text{Switching power}} + \underbrace{Q_{SC} \times V_{DD} \times f \times N_{SW}}_{\text{Short-circ. power}} + \underbrace{I_{leak} \times V_{DD}}_{\text{Leakage power}}$$

Switching power
Power required to charge/discharge circuit nodes

Short-circ. power
Dissipation due to short-circuit current

Leakage power
Dissipation due to leakage current

- Leakage power is consumed even if the circuit is idle (standby). The only way to avoid is decoupling from power.

Sources of Power Dissipation in CMOS Devices

$$P = \underbrace{\frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}}_{\text{Switching power}} + \underbrace{Q_{SC} \times V_{DD} \times f \times N_{SW}}_{\text{Short-circ. power}} + \underbrace{I_{leak} \times V_{DD}}_{\text{Leakage power}}$$

Switching power
Power required to charge/discharge circuit nodes

Short-circ. power
Dissipation due to short-circuit current

Leakage power
Dissipation due to leakage current

- Leakage power is consumed even if the circuit is idle (standby). The only way to avoid is decoupling from power.
- Short circuit power is up to 10% of total.

Sources of Power Dissipation in CMOS Devices

$$P = \underbrace{\frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}}_{\text{dynamic}} + \underbrace{Q_{SC} \times V_{DD} \times f \times N_{SW}}_{\text{dynamic}} + \underbrace{I_{leak} \times V_{DD}}_{\text{static}}$$

<u>Switching power</u>	<u>Short-circ. power</u>	<u>Leakage power</u>
Power required to charge/discharge circuit nodes	Dissipation due to short-circuit current	Dissipation due to leakage current

- Leakage power is consumed even if the circuit is idle (standby). The only way to avoid is decoupling from power.
- Short circuit power can be around 10% of total.
- Switching power is still the main source of power consumption.

Power and Energy Consumption

$$P = \frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}$$

$$E = P \times t = \frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}$$

N_{CY} = number of cycles needed for the particular task.

Power and Energy Consumption

$$P = \frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}$$

$$E = P \times t = \frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}$$

N_{CY} = number of cycles needed for the particular task.

- In certain situations we are concerned about power consumption:
 - heat dissipation, cooling:
 - physical deterioration due to temperature.
- Sometimes we want to reduce total energy consumed:
 - battery life.

Power and Energy Consumption

$$P = \frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}$$

$$E = P \times t = \frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}$$

- Reducing power/energy consumption:
 - Reduce supply voltage

Power and Energy Consumption

$$P = \frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}$$

$$E = P \times t = \frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}$$

- Reducing power/energy consumption:
 - Reduce supply voltage
 - Reduce switching activity

Power and Energy Consumption

$$P = \frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}$$

$$E = P \times t = \frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}$$

- Reducing power/energy consumption:
 - Reduce supply voltage
 - Reduce switching activity
 - Reduce capacitance

Power and Energy Consumption

$$P = \frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}$$

$$E = P \times t = \frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}$$

- Reducing power/energy consumption:

- Reduce supply voltage
- Reduce switching activity
- Reduce capacitance
- Reduce number of cycles

System Level Power/Energy Optimization

- Dynamic techniques: applied at run time.

These techniques are applied at run-time in order to reduce power consumption by exploiting idle or low-workload periods.

- Static techniques: applied at design time.

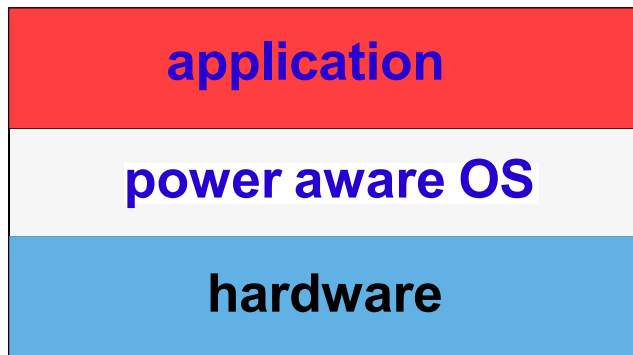
- Compilation for low power: instruction selection considering their power profile, data placement in memory, register allocation.
- Algorithm design: find the algorithm which is the most power-efficient.
- Task mapping and scheduling.

System Level Power/Energy Optimization

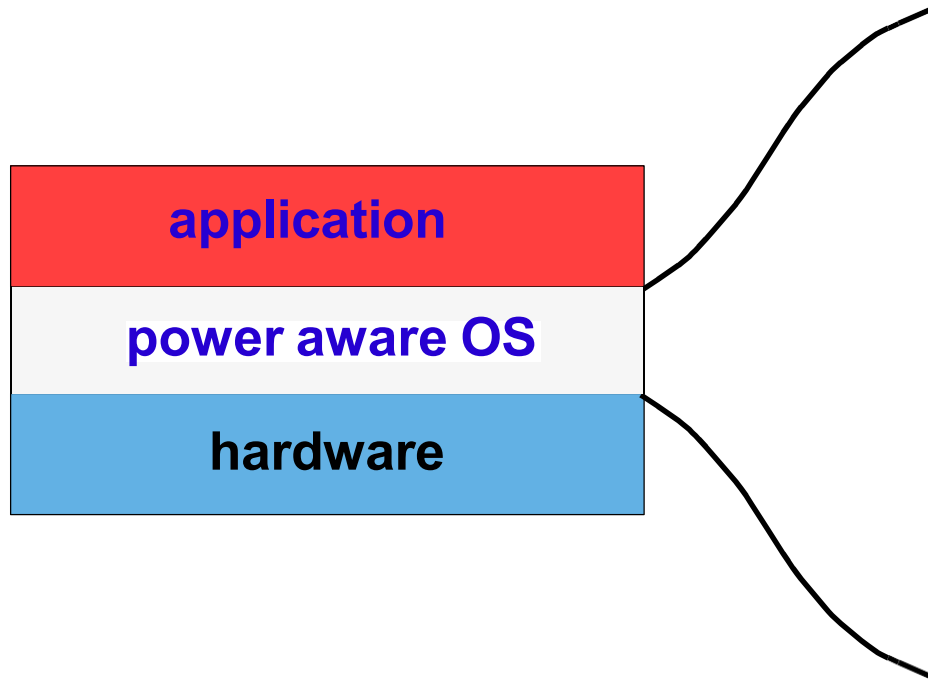
Three techniques will be discussed:

1. Dynamic power management: a dynamic technique.
2. Task mapping: a static technique.
3. Task scheduling with dynamic power scaling: static & dynamic.

Dynamic Power Management (DPM)



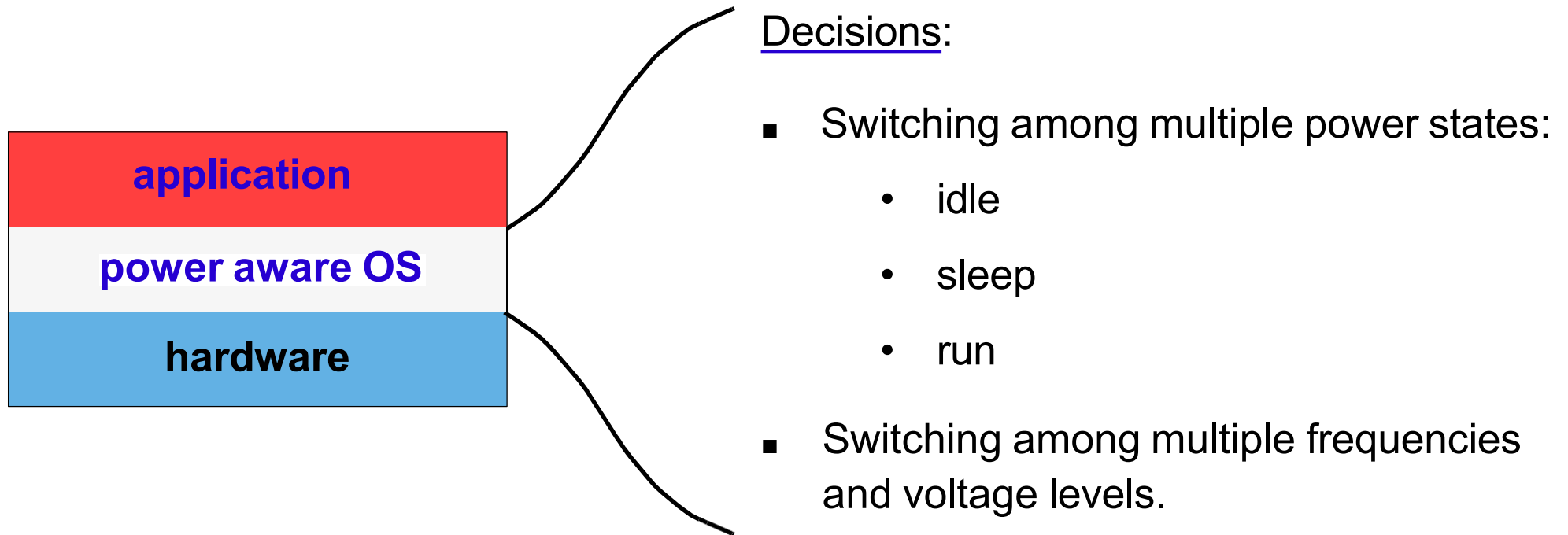
Dynamic Power Management (DPM)



Decisions:

- Switching among multiple power states:
 - idle
 - sleep
 - run
- Switching among multiple frequencies and voltage levels.

Dynamic Power Management (DPM)



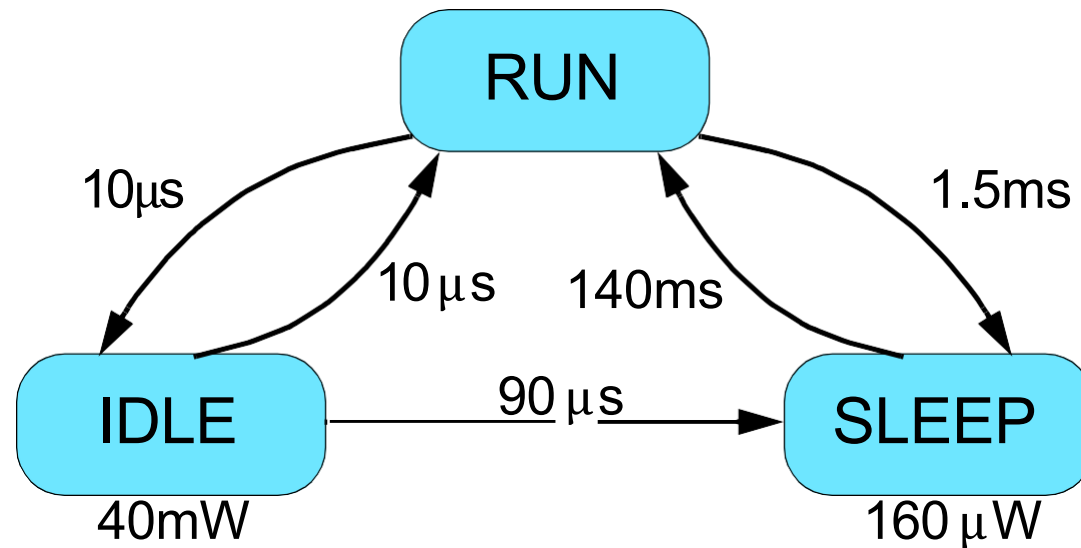
Goal:

- Energy optimization
- QoS constraints satisfied

Dynamic Power Management (DPM)

Intel Xscale Processor

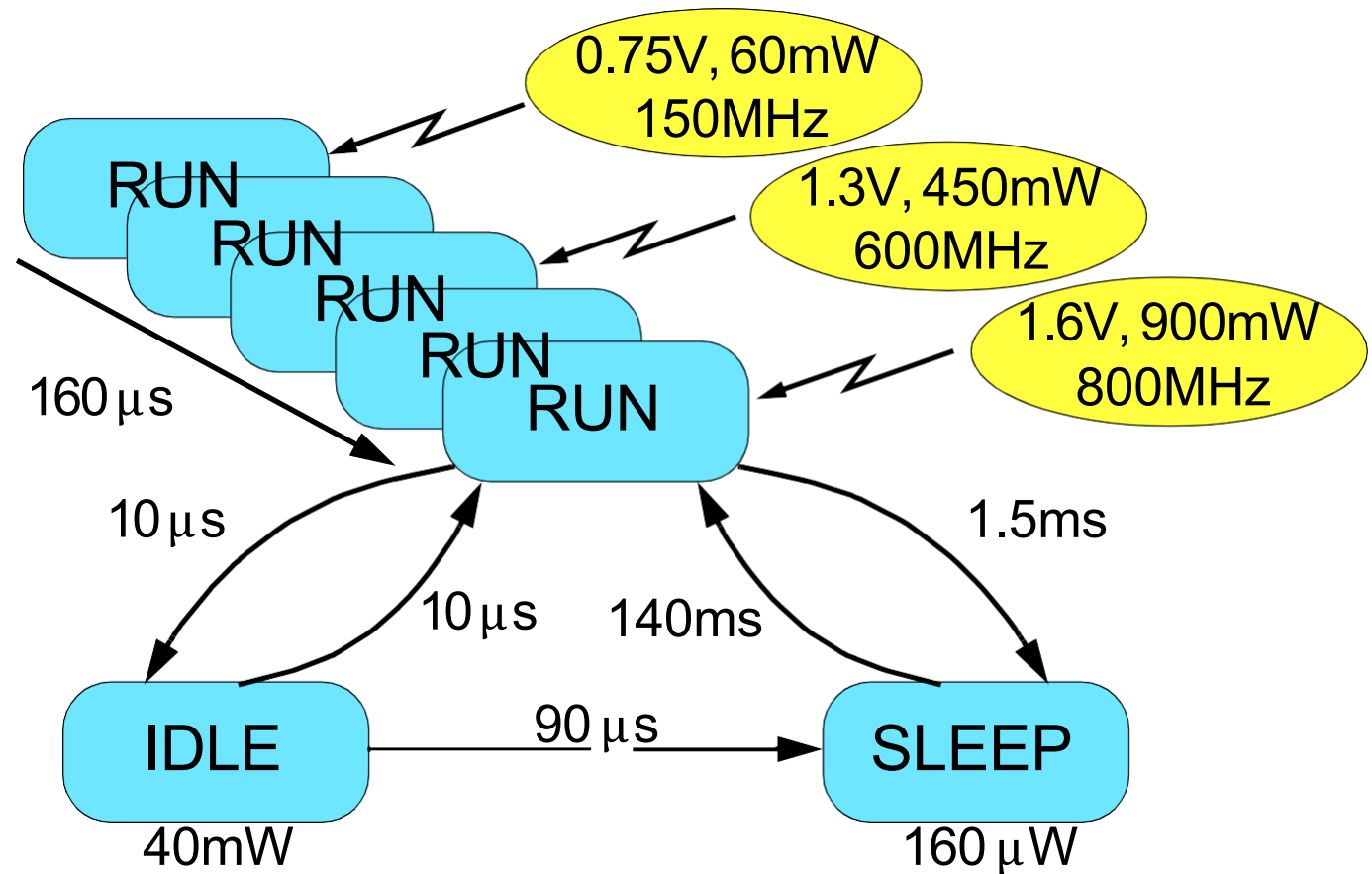
- RUN: operational
- IDLE: Clocks to the CPU are disabled; recovery is through interrupt.
- SLEEP: Mainly powered off; recovery through wake-up event.
- Other intermediate states: DEEP IDLE, STANDBY, DEEP SLEEP



Dynamic Power Management (DPM)

Intel Xscale Processor

- RUN: operational
- IDLE: Clocks to the CPU are disabled; recovery is through interrupt.
- SLEEP: Mainly powered off; recovery through wake-up event.
- Other intermediate states: DEEP IDLE, STANDBY, DEEP SLEEP

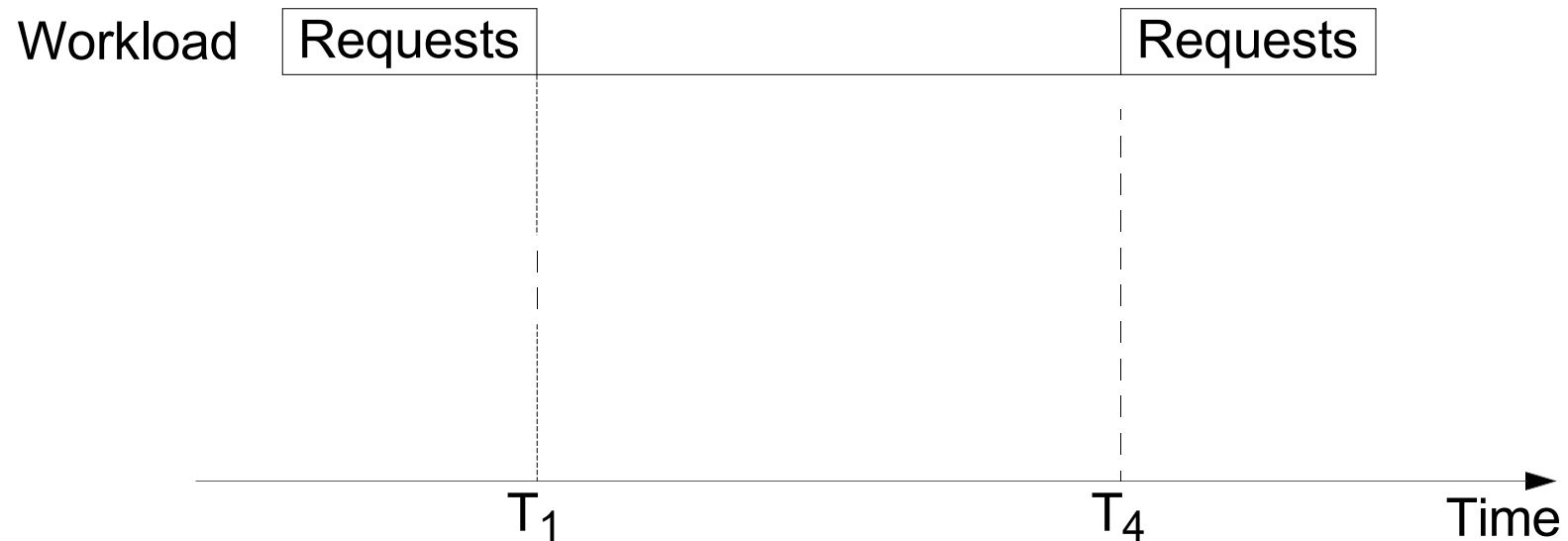


The Basic Concept of DPM

- When there are requests for a device → the device is *busy*; otherwise it is *idle*.
- When the device is idle, it can be shut down to enter a low-power sleeping state.

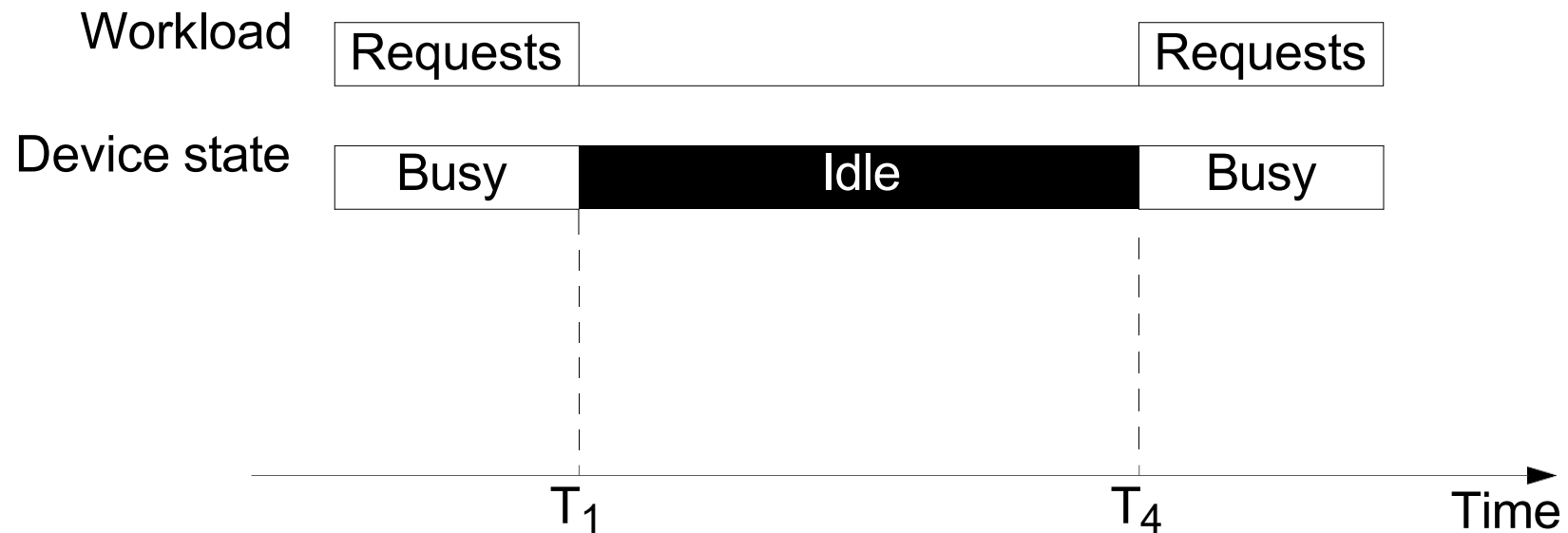
The Basic Concept of DPM

- When there are requests for a device → the device is *busy*; otherwise it is *idle*.
- When the device is idle, it can be shut down to enter a low-power sleeping state.



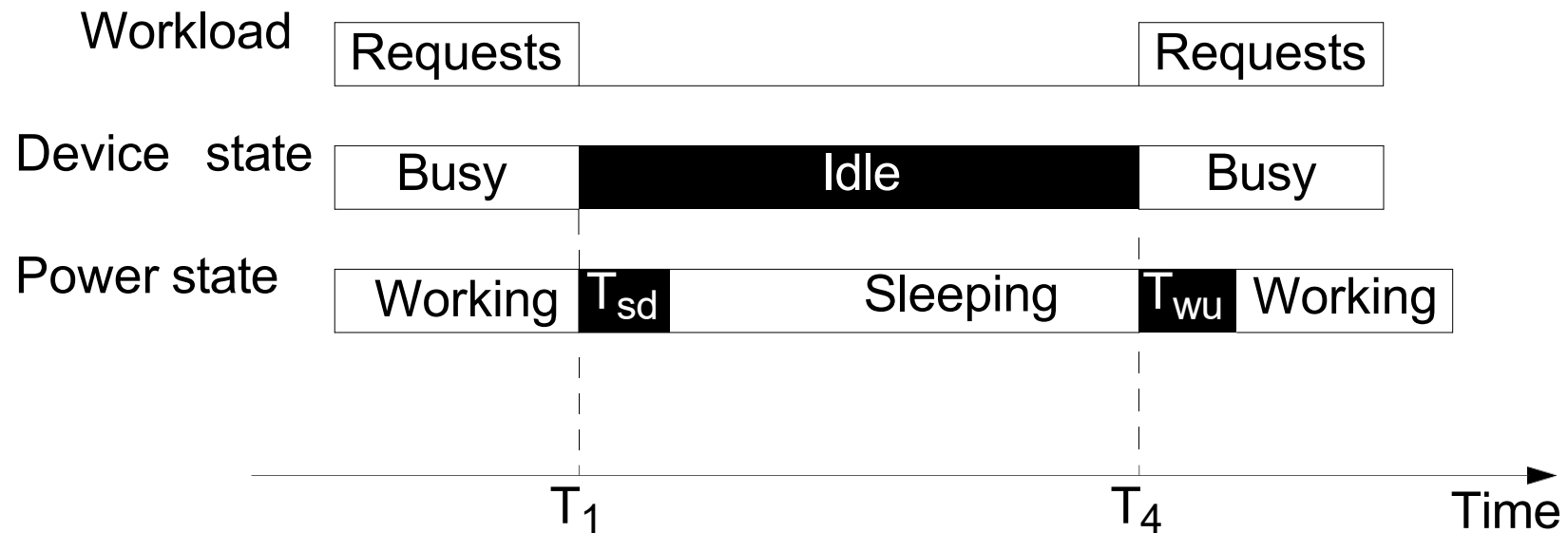
The Basic Concept of DPM

- When there are requests for a device → the device is *busy*; otherwise it is *idle*.
- When the device is idle, it can be shut down to enter a low-power sleeping state.



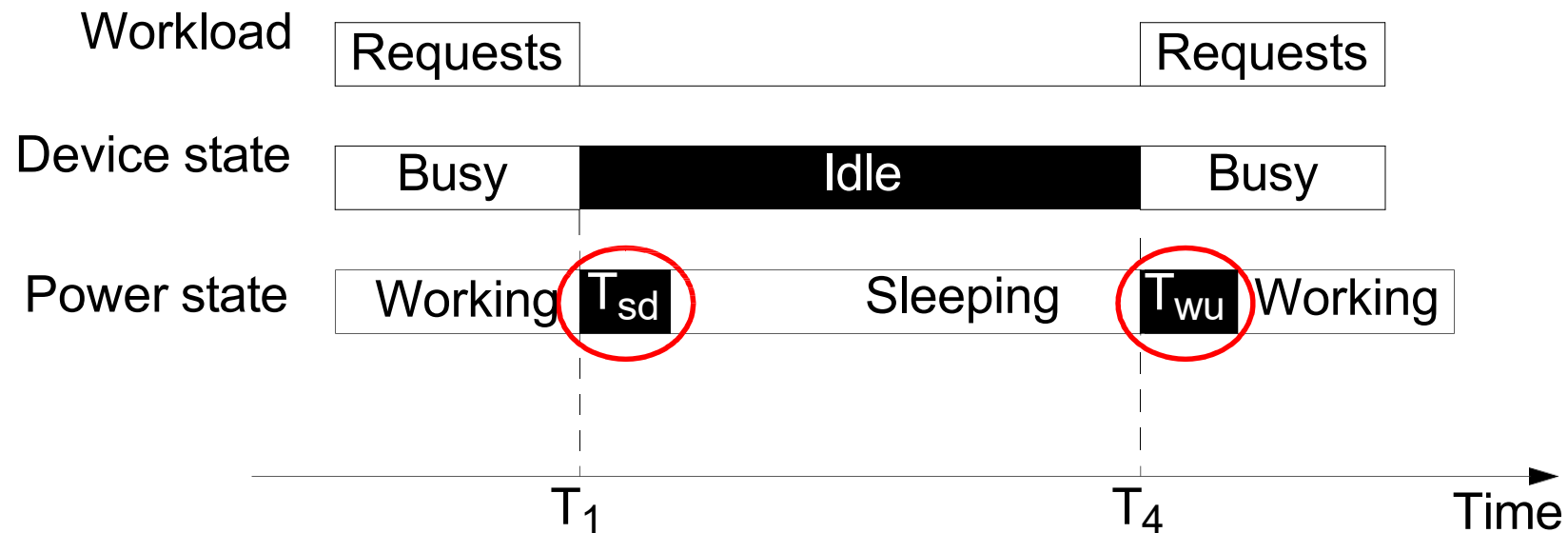
The Basic Concept of DPM

- When there are requests for a device P the device is *busy*; otherwise it is *idle*.
- When the device is idle, it can be shut down to enter a low-power sleeping state.



The Basic Concept of DPM

- When there are requests for a device P the device is *busy*; otherwise it is *idle*.
- When the device is idle, it can be shut down to enter a low-power sleeping state.



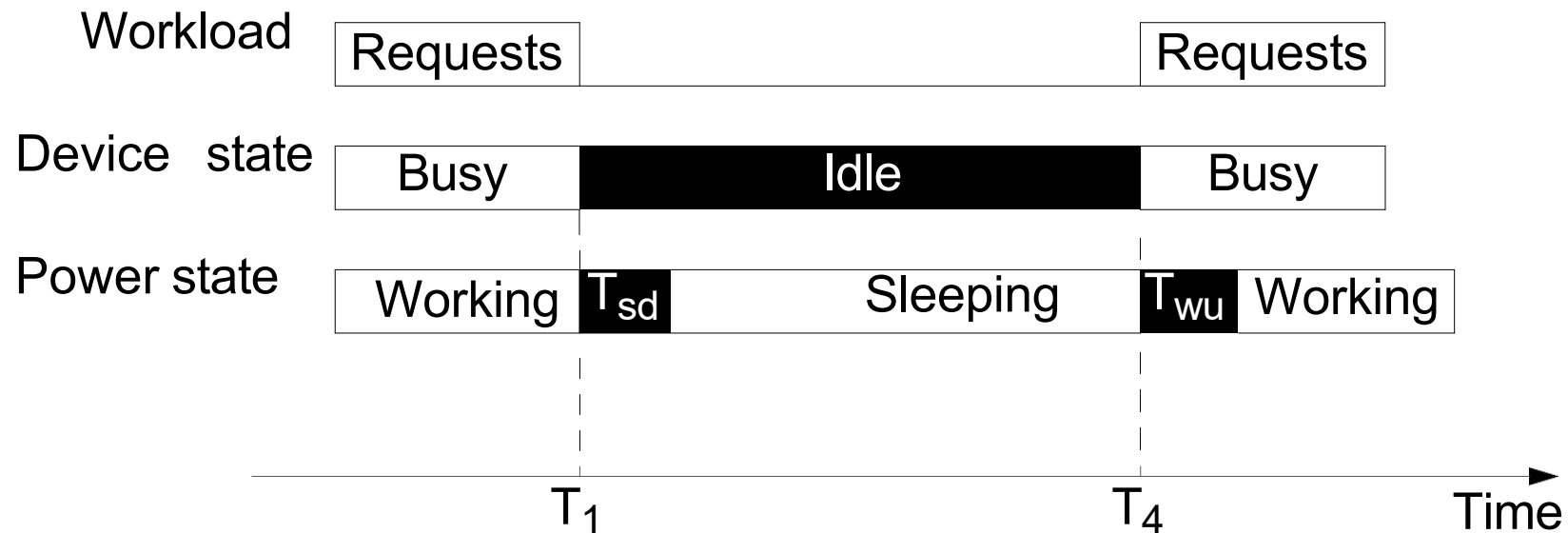
- Changing the power state takes *time and extra energy*.
 - T_{sd} : shutdown delay
 - T_{wu} : wake-up delay



Send the device to sleep only if the saved energy justifies the overhead!

The Basic Concept of DPM

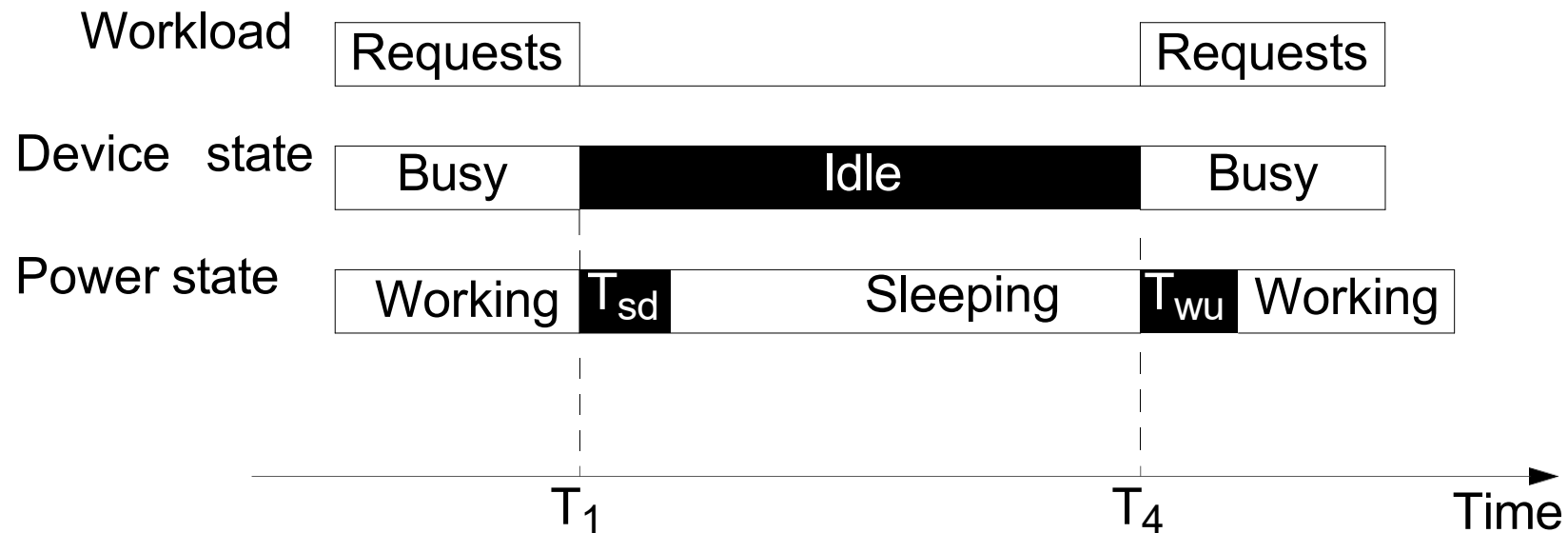
- When there are requests for a device P the device is *busy*; otherwise it is *idle*.
- When the device is idle, it can be shut down to enter a low-power sleeping state.



- The main Problems:
 - Don't shut down such that delays occur too frequently.
 - Don't shut down such that the savings due to the sleeping are smaller than the energy overhead of the state changes.

Power Management Policies

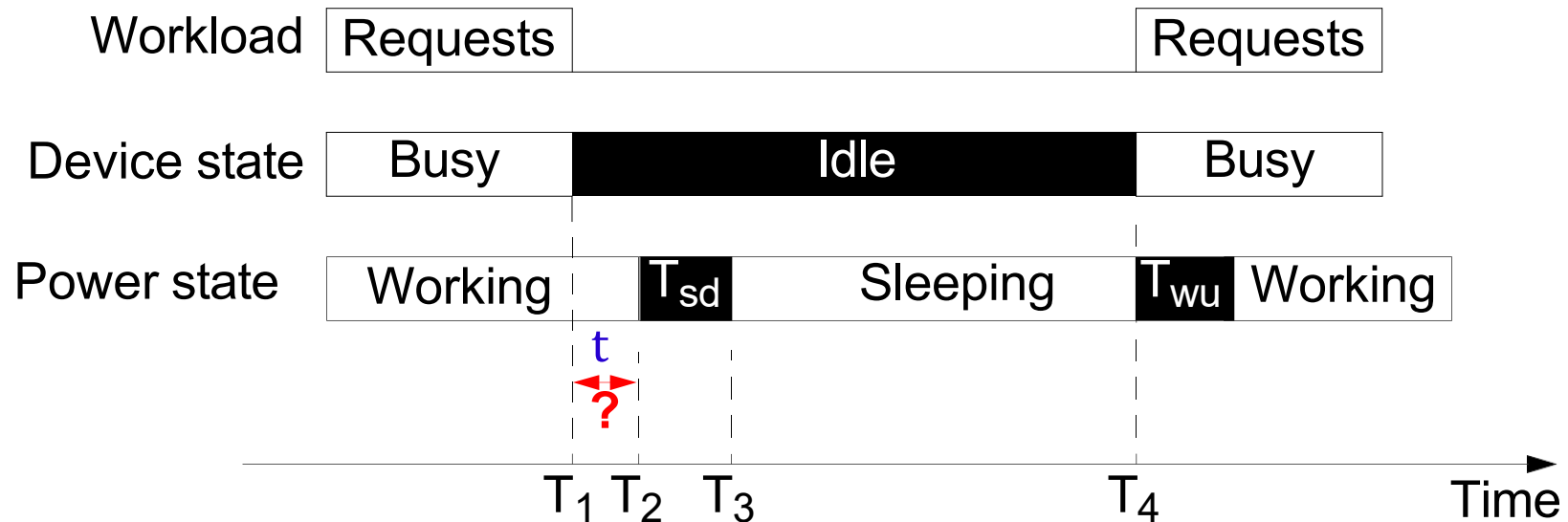
- When there are requests for a device P the device is *busy*; otherwise it is *idle*.
- When the device is idle, it can be shut down to enter a low-power sleeping state.



- Power management policies are concerned with predictions of idle periods:
 - For shut-down: try to predict how long the idle period will be in order to decide if a shut-down should be performed.
 - For wake-up: try to predict when the idle period ends, in order to avoid user delays due to T_{wu} . - Very difficult!

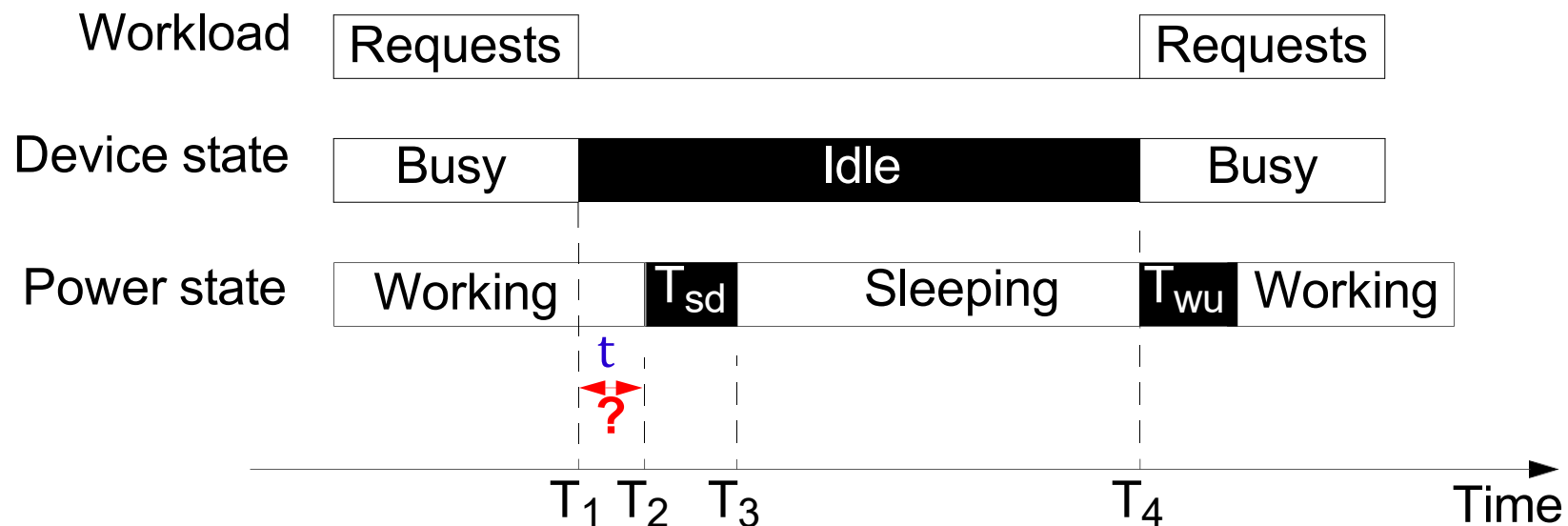
Time-out Policy

- It is assumed that, after a device is idle for a period t , it will stay idle for at least a period which makes it efficient to shut down.



Time-out Policy

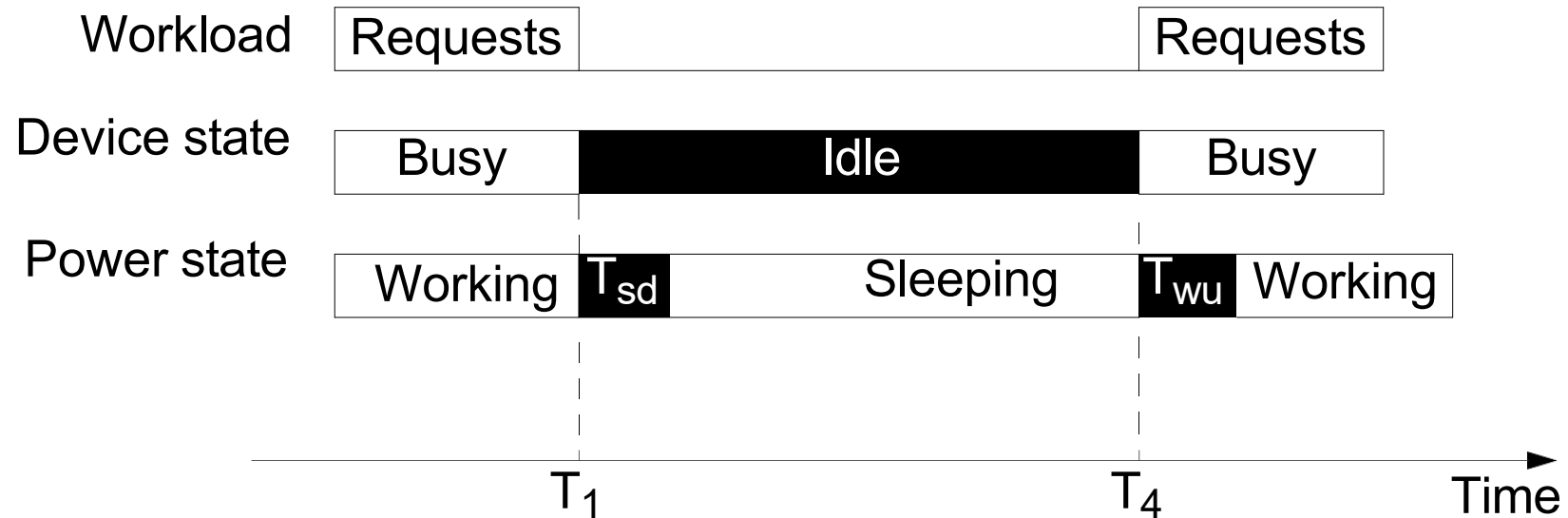
- It is assumed that, after a device is idle for a period t , it will stay idle for at least a period which makes it efficient to shut down.
- Drawback: you waste energy during the period t (compared to instantaneous shut-down).



- Policies:
 - Fixed time-out period: you set the value of t , which stays constant.
 - Adjusted at run-time: increase or decrease t , depending on the length of previous idle periods.

Predictive Policy

- The length of an idle period is predicted. If the predicted idle period is long enough, the shut-down is performed immediately (time interval $t = 0$).



Example: A Very Simple Predictive Policy

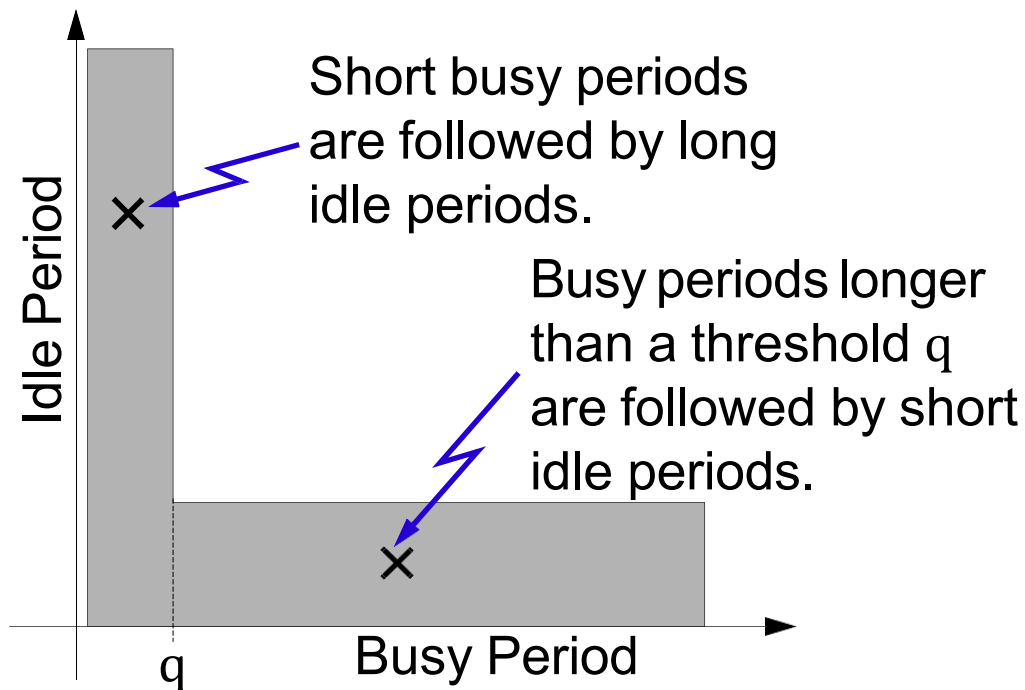
- This is just a *very particular* example! This policy has been proposed for a *very particular* application, after intensive experiments. This policy might not work for any other application!

Example: A Very Simple Predictive Policy

- This is just a *very particular* example! This policy has been proposed for a *very particular* application, after intensive experiments. This policy might not work for any other application!
- Measurements *on the particular application*, have shown an L-shaped distribution for:

Idle Period

 $\text{Previous Busy Period}$

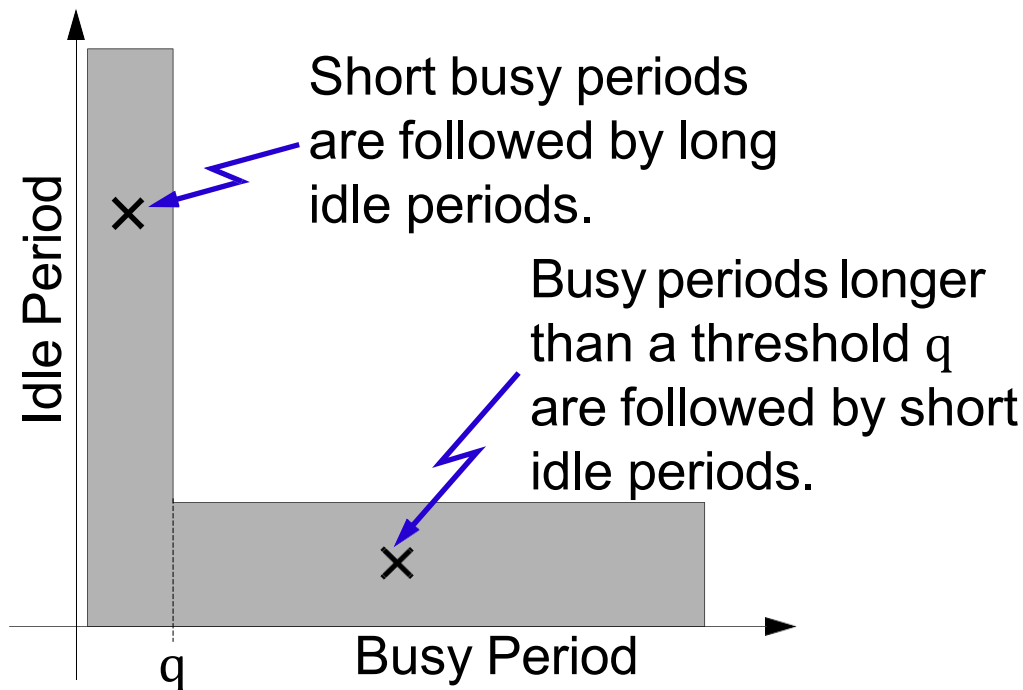


Example: A Very Simple Predictive Policy

- This is just a *very particular* example! This policy has been proposed for a *very particular* application, after intensive experiments. This policy might not work for any other application!
- Measurements *on the particular application*, have shown an L-shaped distribution for:

Idle Period

 $\text{Previous Busy Period}$



Shut down after short ($< q$) busy period!

Advanced Predictive Policies

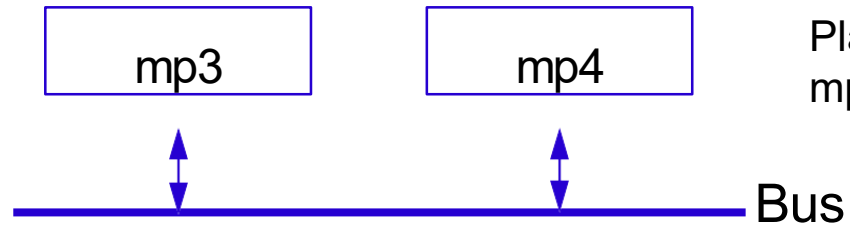
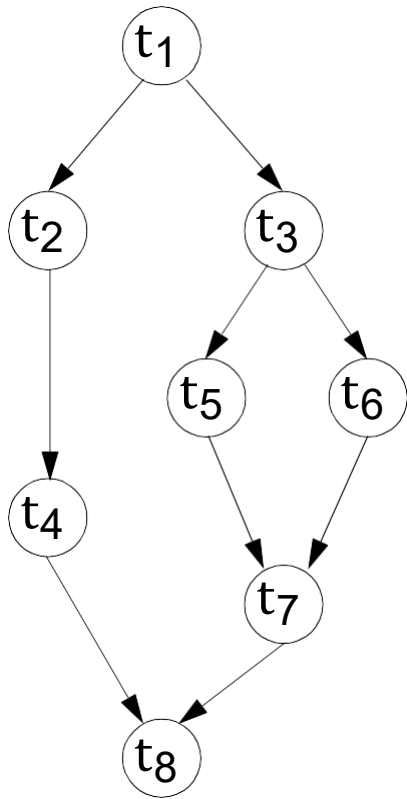
- In the previous example the authors were very lucky!
 - One single application is running on the platform.
 - By profiling, they were able to draw a *very simple* conclusion regarding the run-time behaviour, expressed by that “L-shape” diagram.
- Most often the situation is not that simple:
 - We do not know in advance all application running on the system;
 - The behaviour of the applications changes during run-time, depending on environment and input data.

More advanced run-time prediction techniques have to be applied like e.g. based on statistics, stochastic modelling, and machine learning

Dynamic Power Management (DPM)

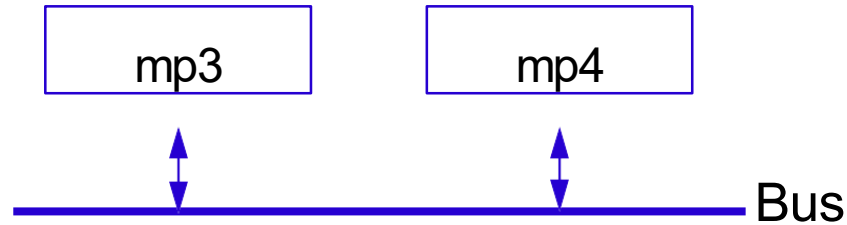
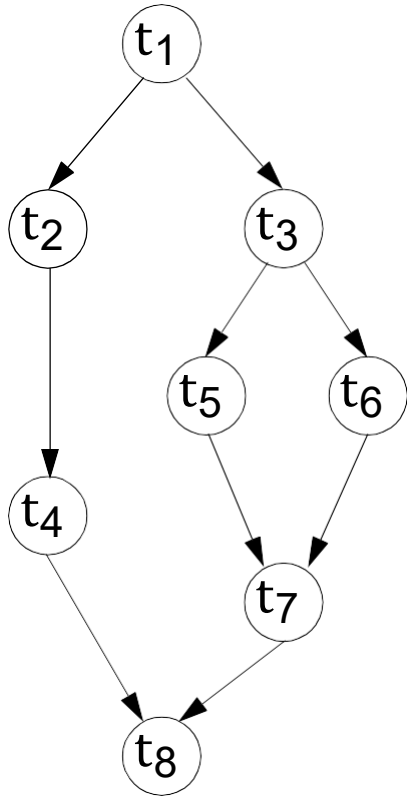
- For many embedded systems DPM techniques, like presented before, are not appropriate:
 - They have time constraints → we have to keep deadlines (usually we cannot afford shut-down and wake-up times).
 - The OS is simple&fast → no sophisticated run-time techniques.
 - The application is known at design time → we know a lot about the application and optimize already at design time.

Mapping for Low Energy



Platform with two microprocessors mp3 and mp4, and a communication bus

Mapping for Low Energy



Consider a mapping:

mp3: t_1, t_3, t_6, t_7, t_8

mp4: t_2, t_4, t_5 .

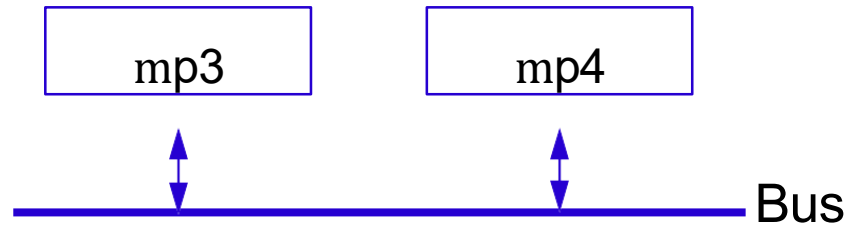
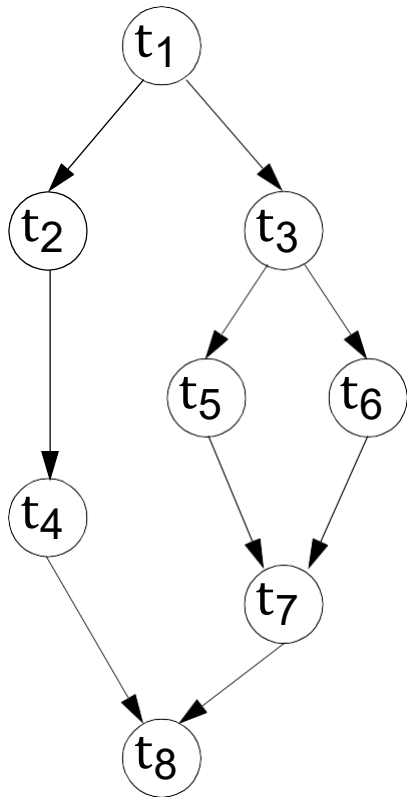
Communication times and energy:

C_{1-2} : $t = 1$; $E = 3$. C_{3-5} : $t = 2$; $E = 5$.

C_{4-8} : $t = 1$; $E = 3$. C_{5-7} : $t = 1$; $E = 3$.

Task	WCET		Energy	
	mp3	mp4	mp3	mp4
t_1	5	6	5	3
t_2	7	9	8	4
t_3	5	6	5	3
t_4	8	10	6	4
t_5	10	11	8	6
t_6	17	21	15	10
t_7	10	14	8	7
t_8	15	19	14	9

Mapping for Low Energy



Consider a mapping:

mp3: t_1, t_3, t_6, t_7, t_8

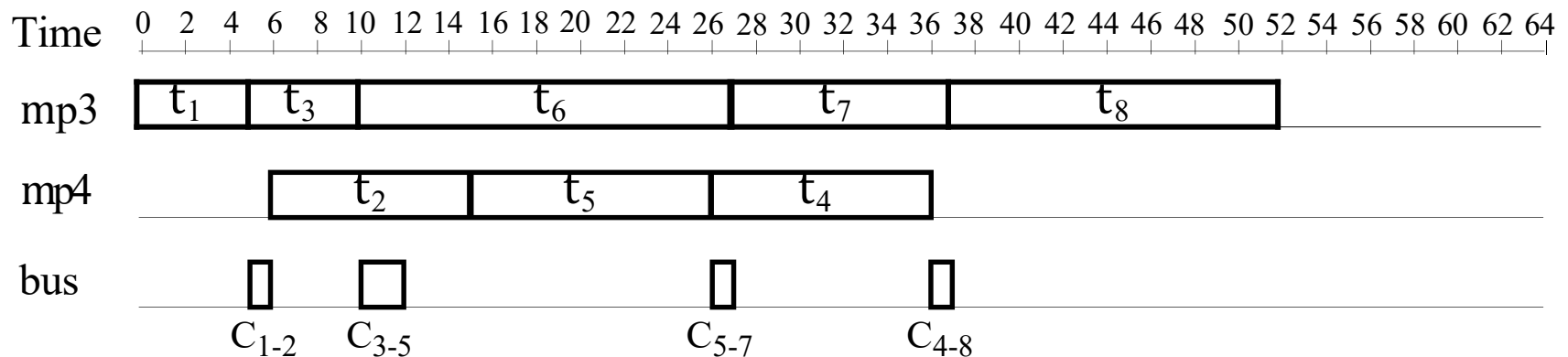
mp4: t_2, t_4, t_5 .

Communication times and energy:

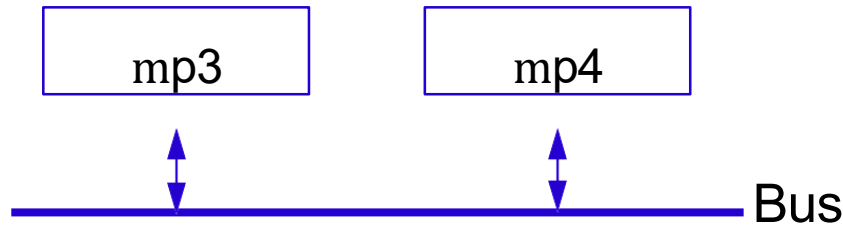
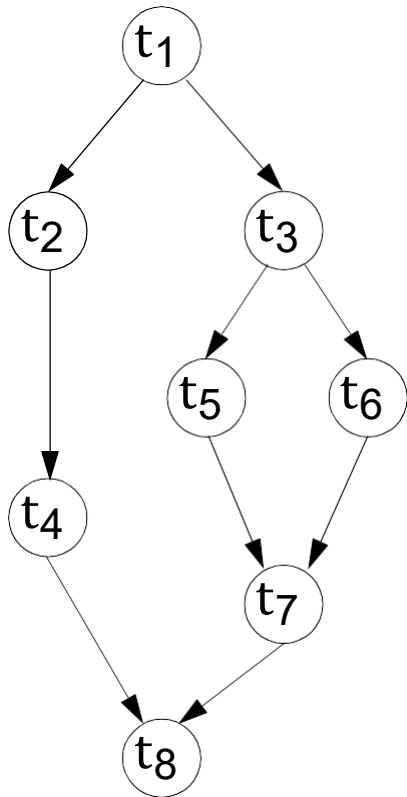
C_{1-2} : $t = 1$; $E = 3$. C_{3-5} : $t = 2$; $E = 5$.

C_{4-8} : $t = 1$; $E = 3$. C_{5-7} : $t = 1$; $E = 3$.

Task	WCET		Energy	
	mp3	mp4	mp3	mp4
t_1	5	6	5	3
t_2	7	9	8	4
t_3	5	6	5	3
t_4	8	10	6	4
t_5	10	11	8	6
t_6	17	21	15	10
t_7	10	14	8	7
t_8	15	19	14	9



Mapping for Low Energy



Consider a mapping:

mp3: t_1, t_3, t_6, t_7, t_8

mp4: t_2, t_4, t_5 .

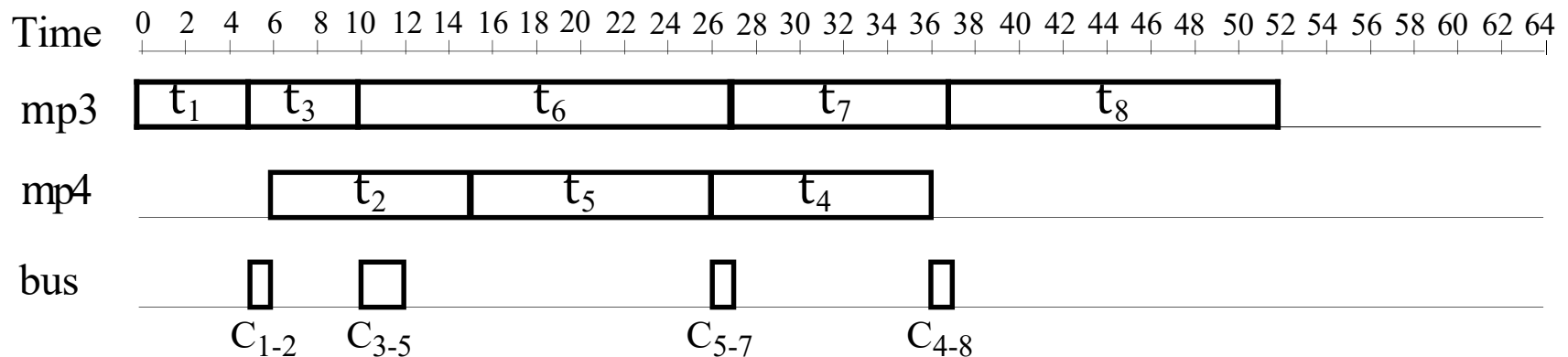
Communication times and energy:

C_{1-2} : $t = 1$; $E = 3$. C_{3-5} : $t = 2$; $E = 5$.

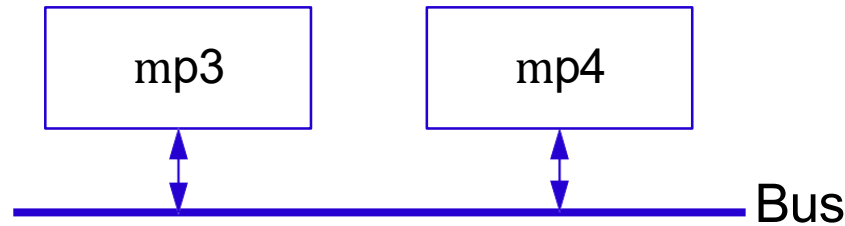
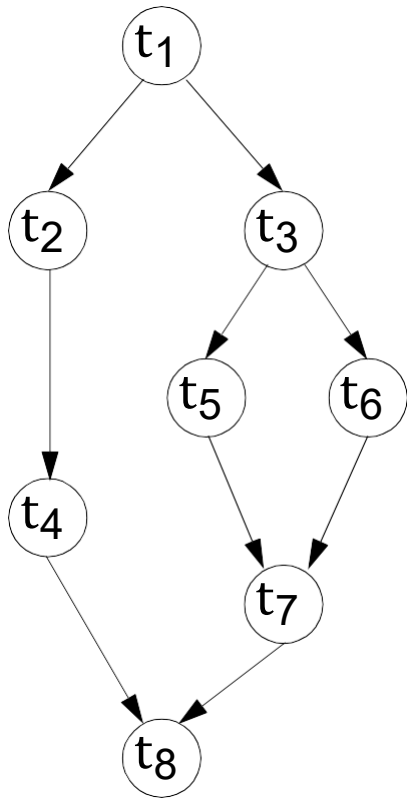
C_{4-8} : $t = 1$; $E = 3$. C_{5-7} : $t = 1$; $E = 3$.

Task	WCET		Energy	
	mp3	mp4	mp3	mp4
t_1	5	6	5	3
t_2	7	9	8	4
t_3	5	6	5	3
t_4	8	10	6	4
t_5	10	11	8	6
t_6	17	21	15	10
t_7	10	14	8	7
t_8	15	19	14	9

Execution time: 52; Energy consumed: 75



Mapping for Low Energy



Consider another mapping:

mp3: t_1, t_3, t_6, t_7, t_8

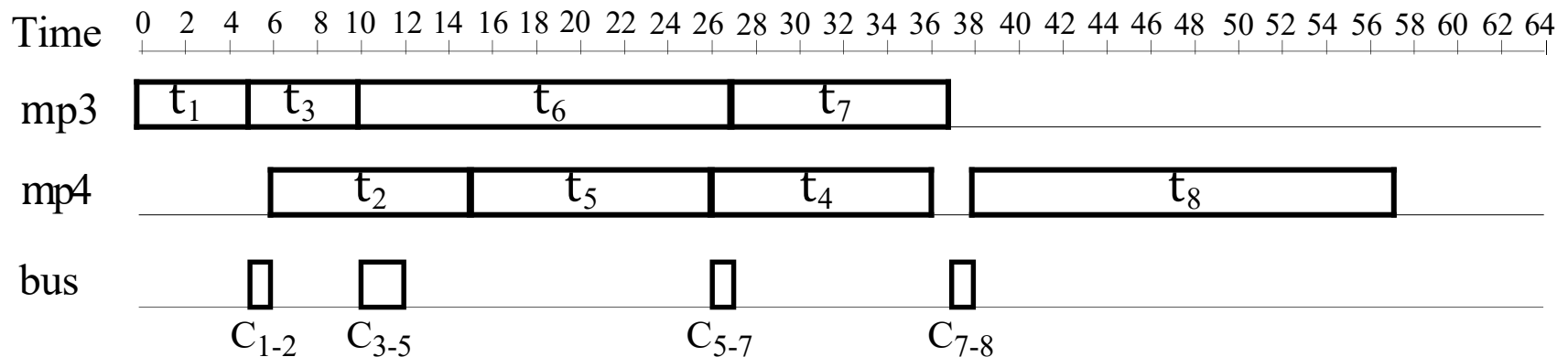
mp4: t_2, t_4, t_5, t_8

Communication times and energy:

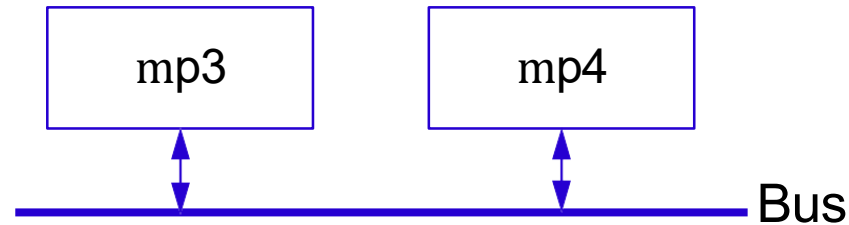
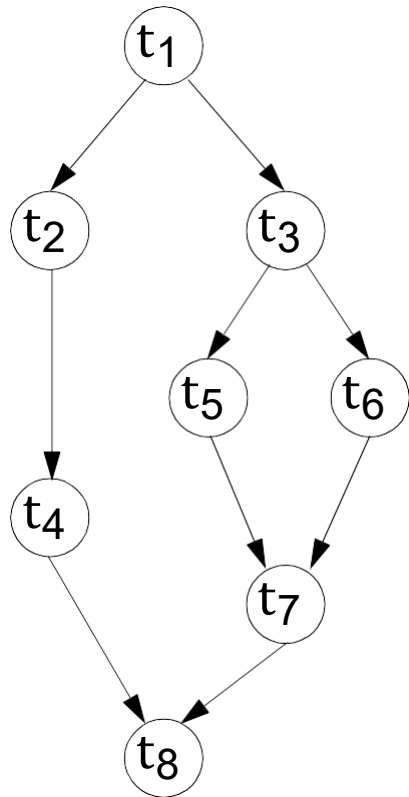
C_{1-2} : $t = 1$; $E = 3$. C_{3-5} : $t = 2$; $E = 5$.

C_{7-8} : $t = 1$; $E = 3$. C_{5-7} : $t = 1$; $E = 3$.

Task	WCET		Energy	
	mp3	mp4	mp3	mp4
t_1	5	6	5	3
t_2	7	9	8	4
t_3	5	6	5	3
t_4	8	10	6	4
t_5	10	11	8	6
t_6	17	21	15	10
t_7	10	14	8	7
t_8	15	19	14	9



Mapping for Low Energy



Consider a mapping:

mp3: t_1, t_3, t_6, t_7

mp4: t_2, t_4, t_5, t_8

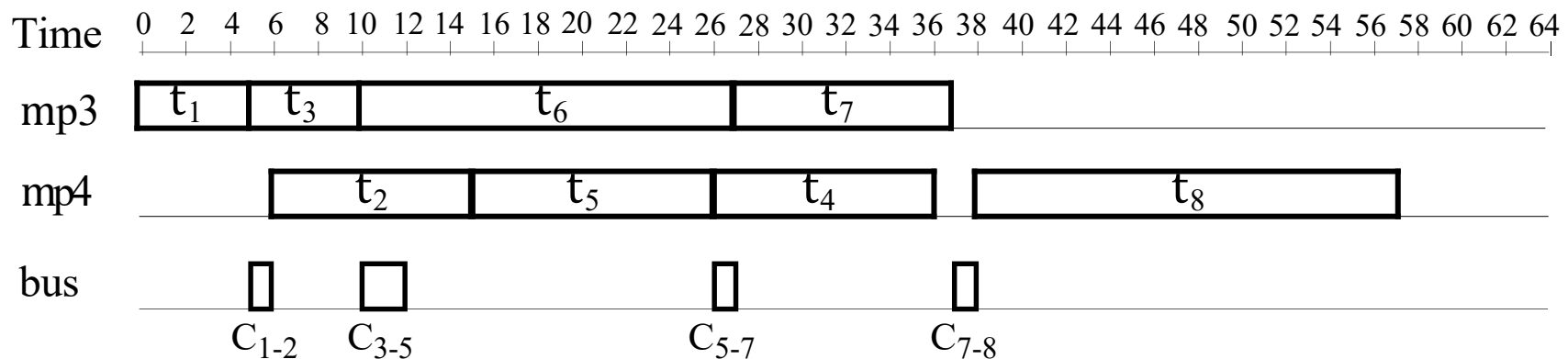
Communication times and energy:

C_{1-2} : $t = 1$; $E = 3$. C_{3-5} : $t = 2$; $E = 5$.

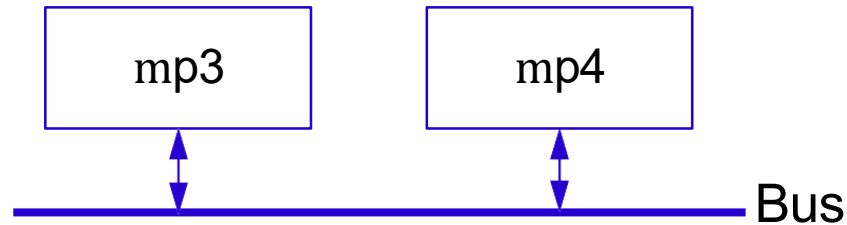
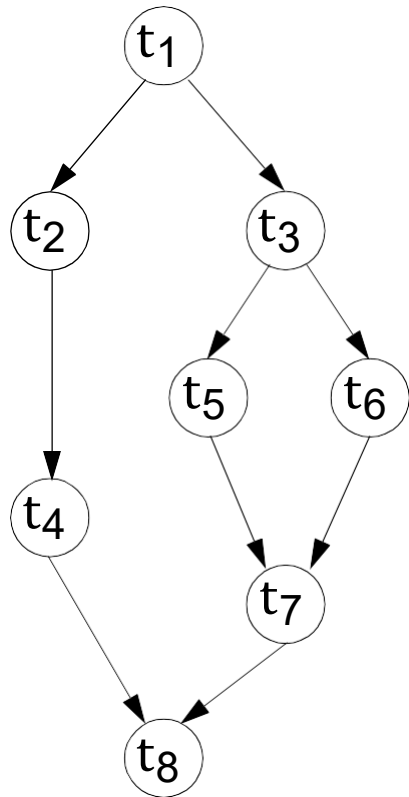
C_{7-8} : $t = 1$; $E = 3$. C_{5-7} : $t = 1$; $E = 3$.

Task	WCET		Energy	
	mp3	mp4	mp3	mp4
t_1	5	6	5	3
t_2	7	9	8	4
t_3	5	6	5	3
t_4	8	10	6	4
t_5	10	11	8	6
t_6	17	21	15	10
t_7	10	14	8	7
t_8	15	19	14	9

Execution time: 57; Energy consumed: 70



Mapping for Low Energy



Task	WCET		Energy	
	mp3	mp4	mp3	mp4
t ₁	5	6	5	3
t ₂	7	9	8	4
t ₃	5	6	5	3
t ₄	8	10	6	4
t ₅	10	11	8	6
t ₆	17	21	15	10
t ₇	10	14	8	7
t ₈	15	19	14	9

- The second mapping with t₈ on mp4 consumes less energy;
 - Assume that we have a maximum allowed delay = 60.



This second mapping is preferable, even if it is slower!

Real-Time Scheduling with Dynamic Voltage Scaling

- The energy consumed by a task, due to switching power:

$$E = \frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}$$

N_{SW} = number of gate transitions per clock cycle.

N_{CY} = number of cycles needed for the task.

Real-Time Scheduling with Dynamic Voltage Scaling

- The energy consumed by a task, due to switching power:

$$E = \frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}$$

N_{SW} = number of gate transitions per clock cycle.

N_{CY} = number of cycles needed for the task.

- Reducing supply voltage V_{DD} is the efficient way to reduce energy consumption.
 - The frequency at which the processor can be operated depends on V_{DD} :

$$f = k \times \frac{(V_{DD} - V_t)^2}{V_{DD}}, \quad k: \text{circuit dependent constant}; V_t: \text{threshold voltage}.$$

Real-Time Scheduling with Dynamic Voltage Scaling

- The energy consumed by a task, due to switching power:

$$E = \frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}$$

N_{SW} = number of gate transitions per clock cycle.

N_{CY} = number of cycles needed for the task.

- Reducing supply voltage V_{DD} is the efficient way to reduce energy consumption.

- The frequency at which the processor can be operated depends on V_{DD} :

$$f = k \times \frac{(V_{DD} - V_t)^2}{V_{DD}}, \text{ } k: \text{circuit dependent constant; } V_t: \text{threshold voltage.}$$

- The execution time of the task:

$$t_{exe} = N_{CY} \times \frac{V_{DD}}{k \times (V_{DD} - V_t)^2}$$

Depends on V_{DD} !

Real-Time Scheduling with Dynamic Voltage Scaling

- The (classical) scheduling problem:

Which task to execute at a certain moment on a certain processor so that time constraints are fulfilled?

Real-Time Scheduling with Dynamic Voltage Scaling

- The (classical) scheduling problem:

Which task to execute at a certain moment on a certain processor so that time constraints are fulfilled?

- The scheduling problem with voltage scaling:

Which task to execute at a certain moment on a certain processor, *and at which voltage level*, so that time constraints are fulfilled and *energy consumption is minimised*?

Real-Time Scheduling with Dynamic Voltage Scaling

- The (classical) scheduling problem:

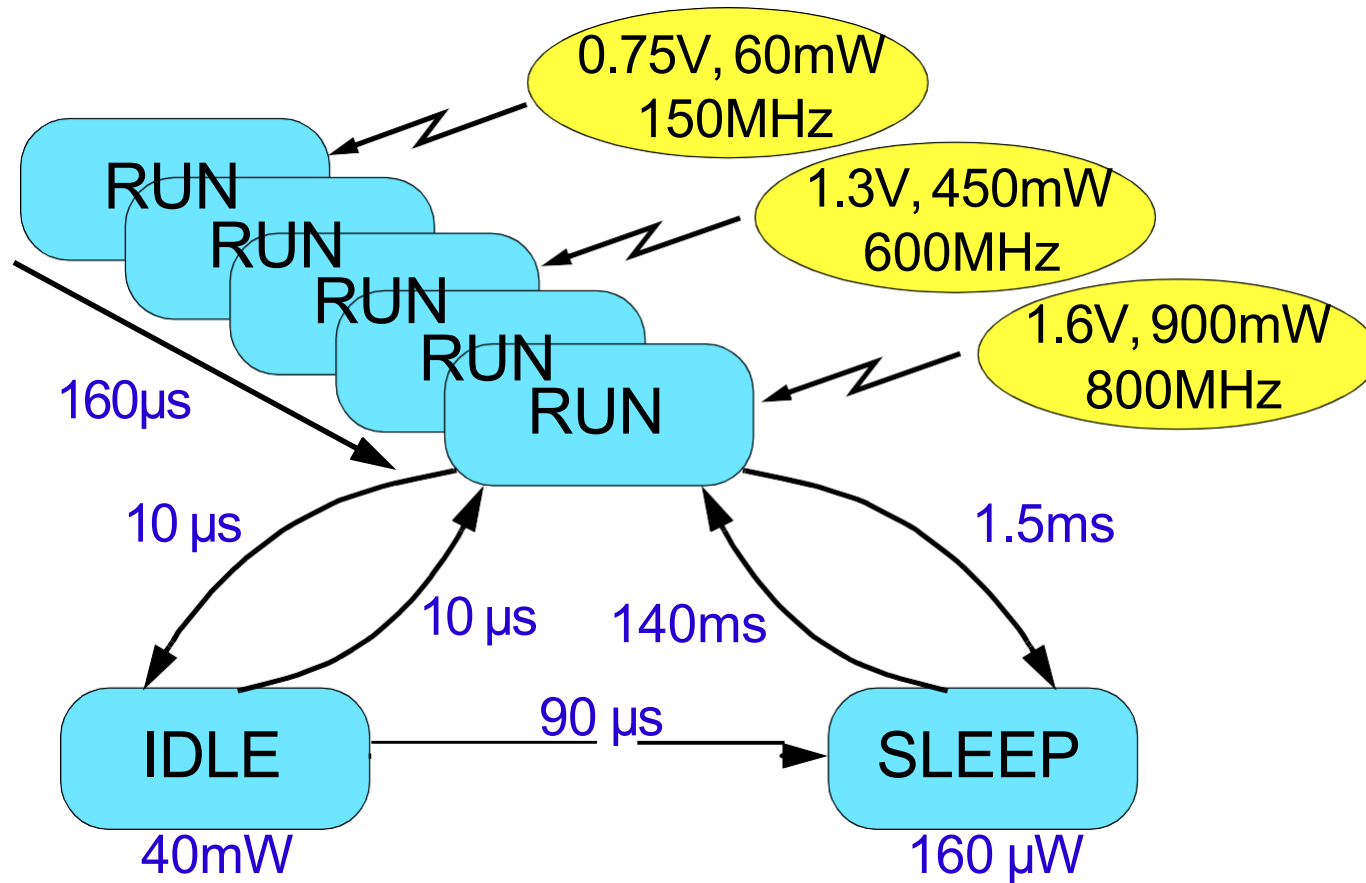
Which task to execute at a certain moment on a certain processor so that time constraints are fulfilled?

- The scheduling problem with voltage scaling:

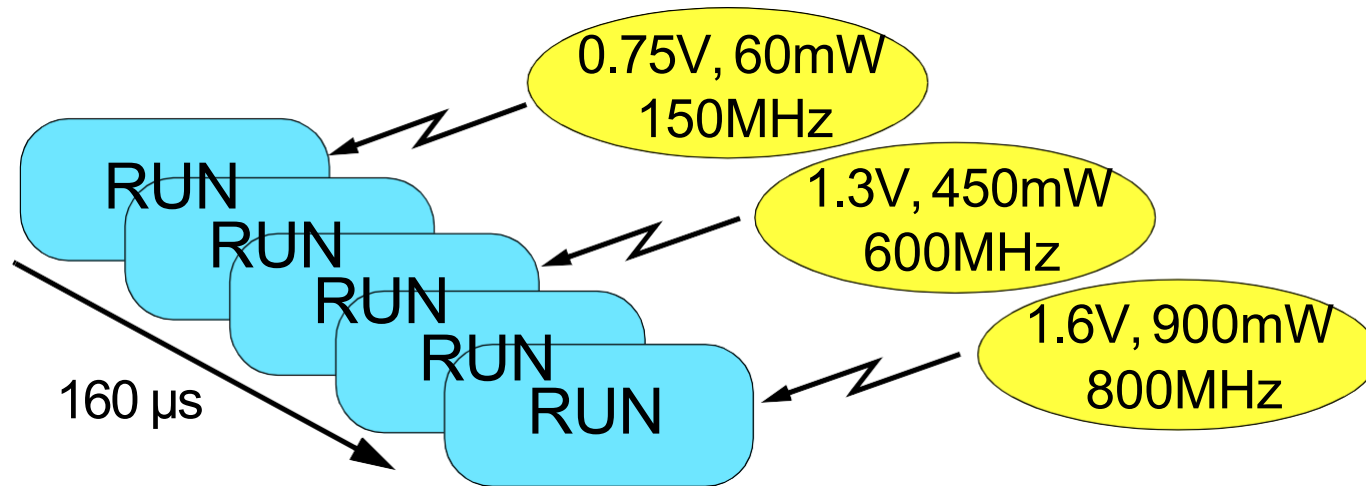
Which task to execute at a certain moment on a certain processor, *and at which voltage level*, so that time constraints are fulfilled and *energy consumption is minimised*?

- The problem: reducing supply voltage extends execution time!

Variable Voltage Processors



Variable Voltage Processors



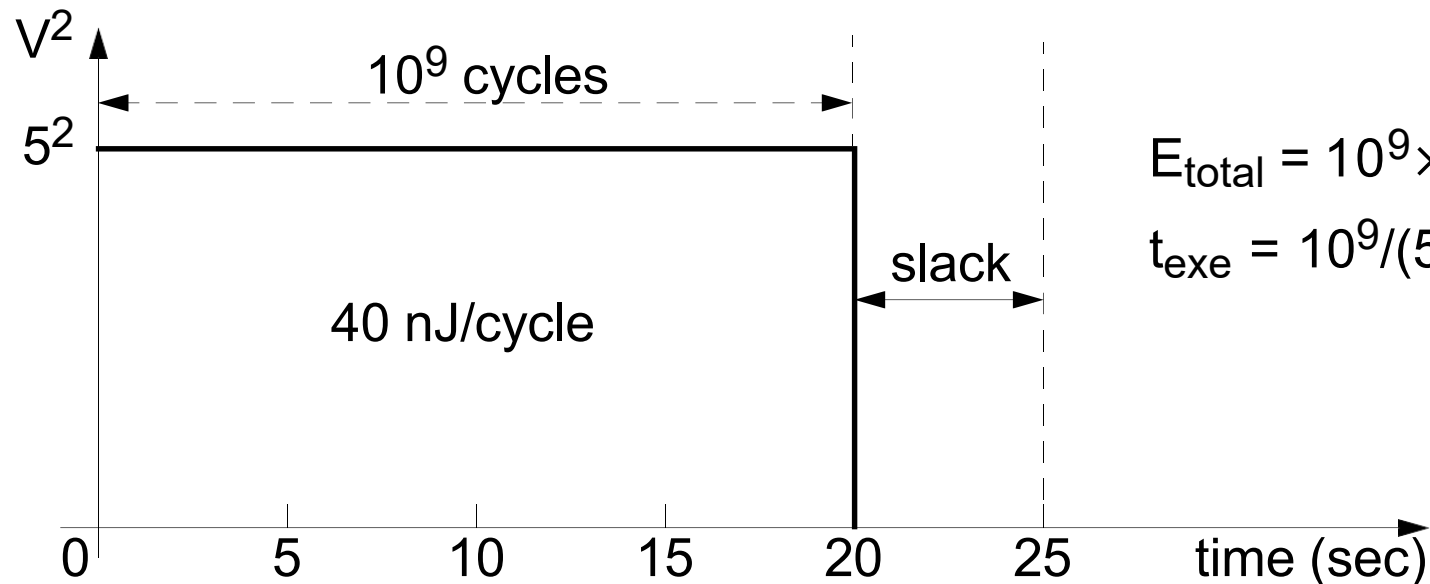
- Several supply voltage levels are available.
- Supply voltage can be changed during run-time.
- Frequency is adjusted to the current supply voltage.

The Basic Principle

- We consider a single task t :
 - total computation: 10^9 execution cycles.
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - energy: 40 nJ/cycle at nominal voltage.
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage.

The Basic Principle

- We consider a single task t:
 - total computation: 10^9 execution cycles.
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - energy: 40 nJ/cycle at nominal voltage.
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage.

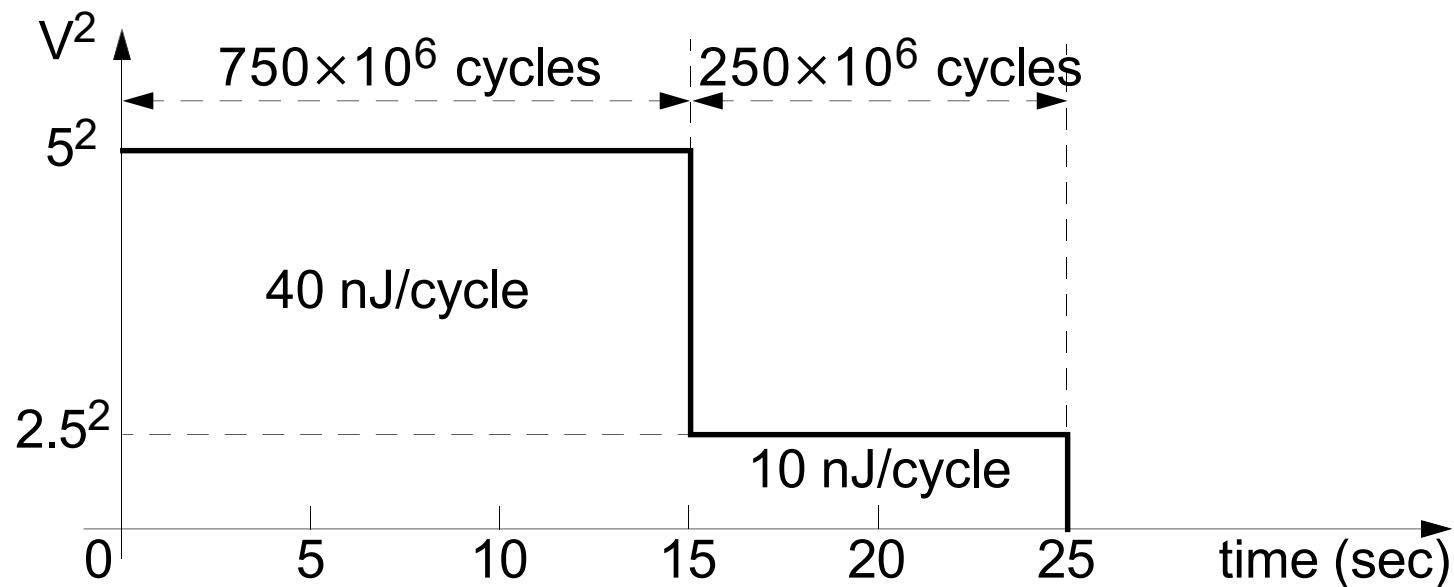


$$E_{\text{total}} = 10^9 \times (40 \times 10^{-9}) = 40 \text{ J}$$

$$t_{\text{exe}} = 10^9 / (50 \times 10^6) = 20 \text{ sec}$$

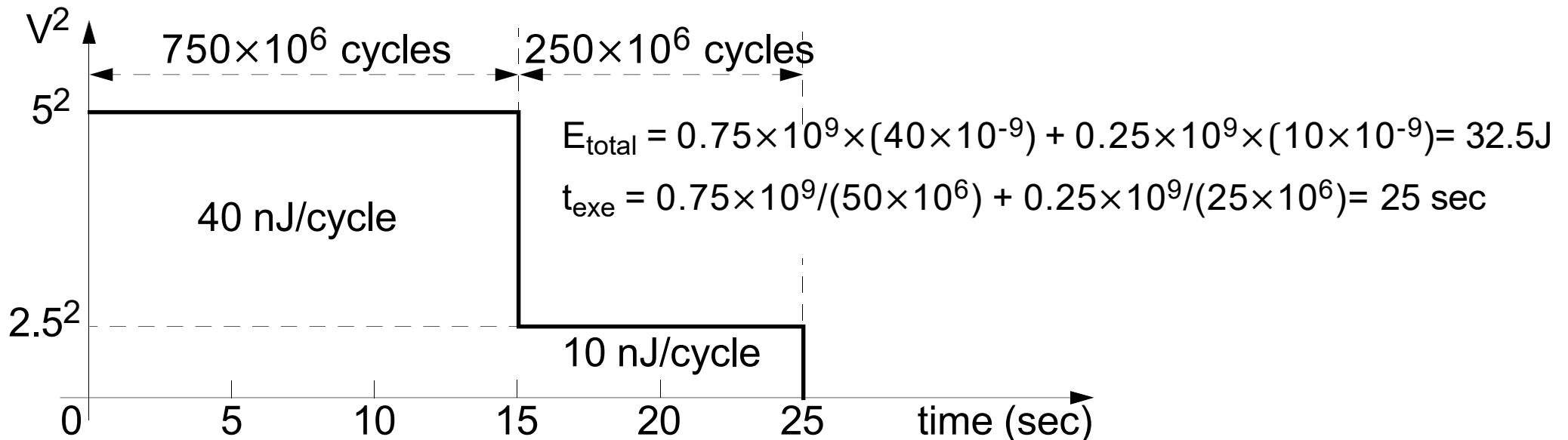
The Basic Principle

- We consider a single task t :
 - total computation: 10^9 execution cycles.
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - energy: 40 nJ/cycle at nominal voltage; at 2.5V: $40 \times 2.5^2 / 5^2 = 10$ nJ/cycle
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage;
at 2.5V: $50 \times 2.5 / 5 = 25$ MHz (25×10^6 cycles/sec).



The Basic Principle

- We consider a single task t:
 - total computation: 10^9 execution cycles.
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - energy: 40 nJ/cycle at nominal voltage; at 2.5V: $40 \times 2.5^2 / 5^2 = 10 \text{ nJ/cycle}$
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage;
at 2.5V: $50 \times 2.5 / 5 = 25 \text{ MHz}$ (25×10^6 cycles/sec).



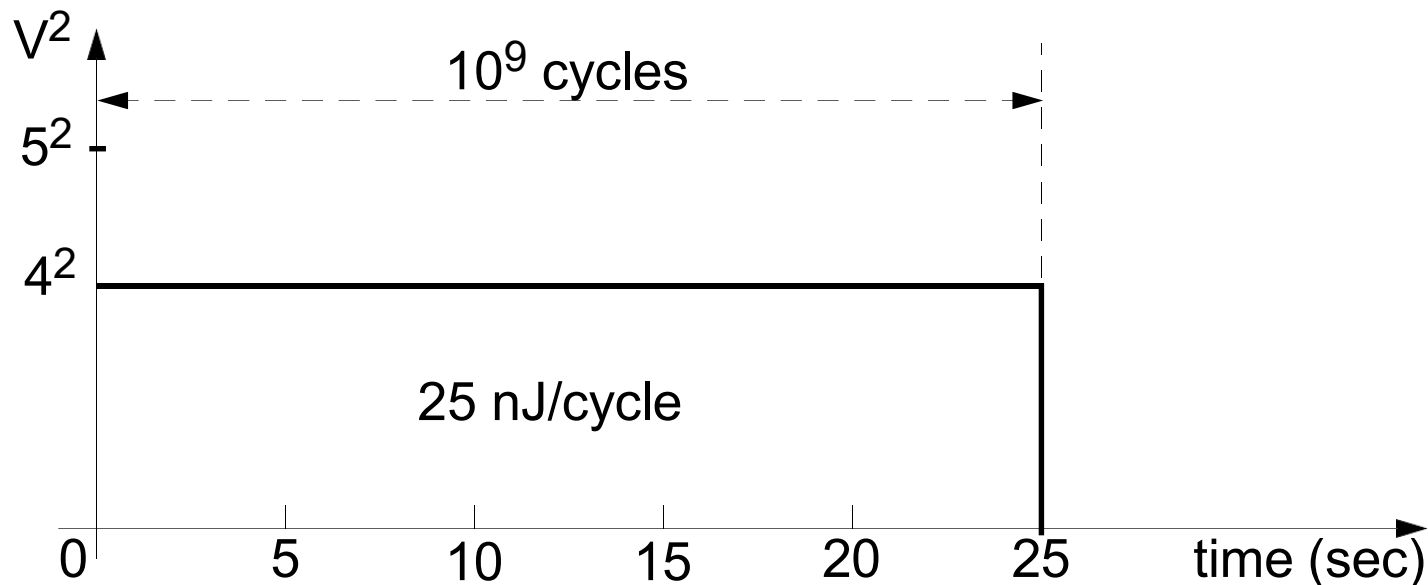
The Basic Principle

- We consider a single task t :
 - total computation: 10^9 execution cycles.
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - energy: 40 nJ/cycle at nominal voltage; at 2.5V: $40 \times 2.5^2 / 5^2 = 10 \text{ nJ/cycle}$
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage;
at 2.5V: $50 \times 2.5 / 5 = 25 \text{ MHz}$ (25×10^6 cycles/sec).

Let's try a different solution!

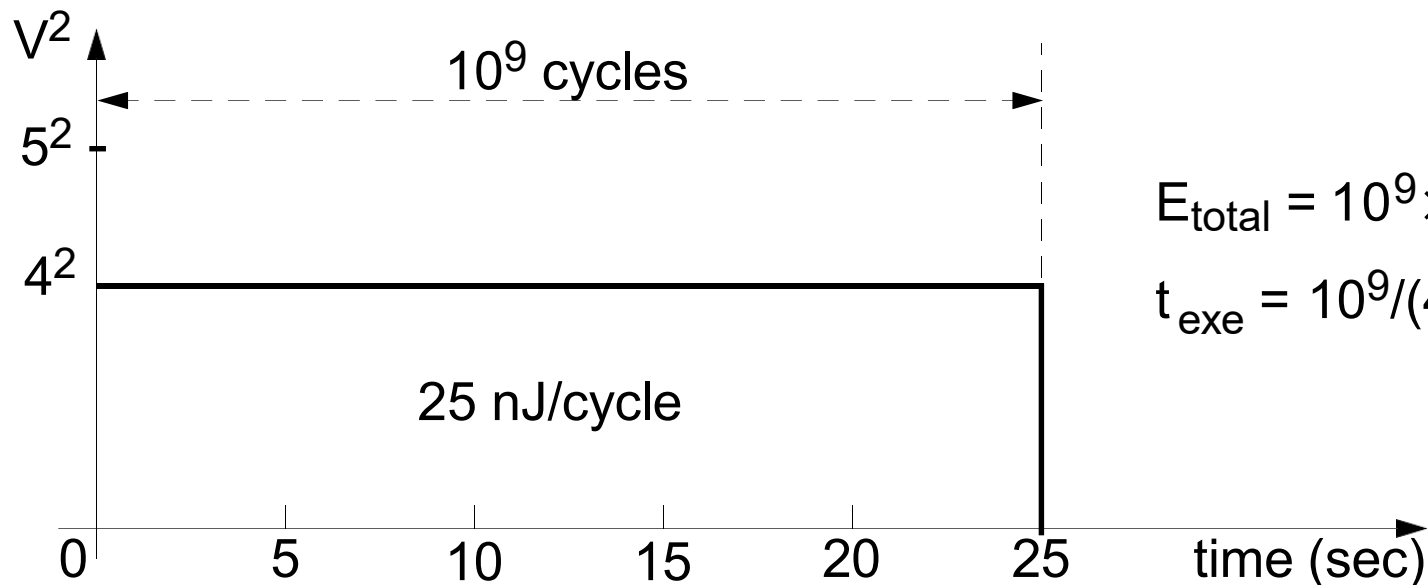
The Basic Principle

- We consider a single task t:
 - total computation: 10^9 execution cycles.
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - energy: 40 nJ/cycle at nominal voltage; at 4V: $40 \times 4^2 / 5^2 = 25 \text{ nJ/cycle}$
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage; at 4V: $50 \times 4 / 5 = 40 \text{ MHz}$ (40×10^6 cycles/sec).



The Basic Principle

- We consider a single task t:
 - total computation: 10^9 execution cycles.
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - energy: 40 nJ/cycle at nominal voltage; **at 4V: $40 \times 4^2 / 5^2 = 25 \text{ nJ/cycle}$**
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage;
at 4V: $50 \times 4 / 5 = 40 \text{ MHz}$ (40×10^6 cycles/sec).

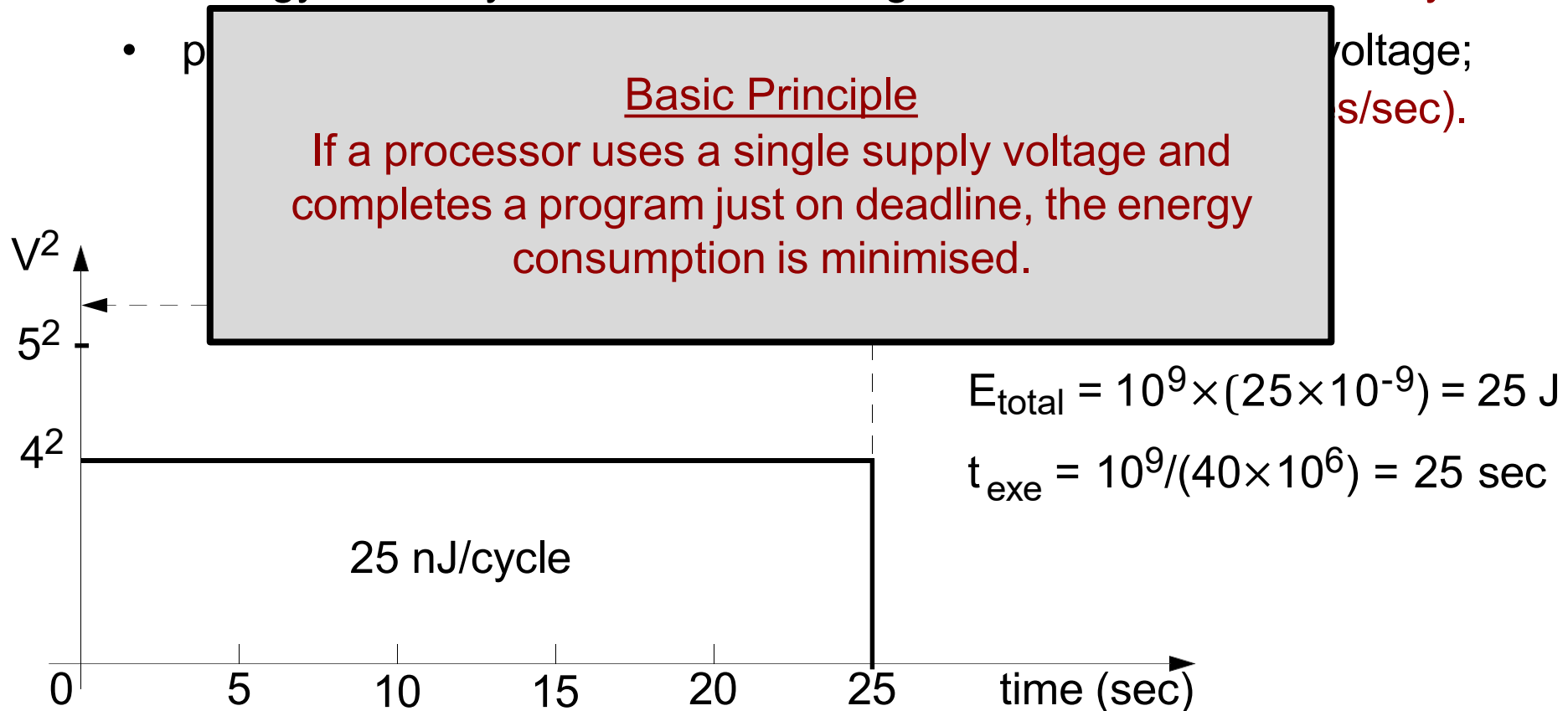


$$E_{\text{total}} = 10^9 \times (25 \times 10^{-9}) = 25 \text{ J}$$

$$t_{\text{exe}} = 10^9 / (40 \times 10^6) = 25 \text{ sec}$$

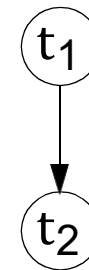
The Basic Principle

- We consider a single task t:
 - total computation: 10^9 execution cycles.
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - energy: 40 nJ/cycle at nominal voltage; at 4V: $40 \times 4^2 / 5^2 = 25 \text{ nJ/cycle}$
 - p



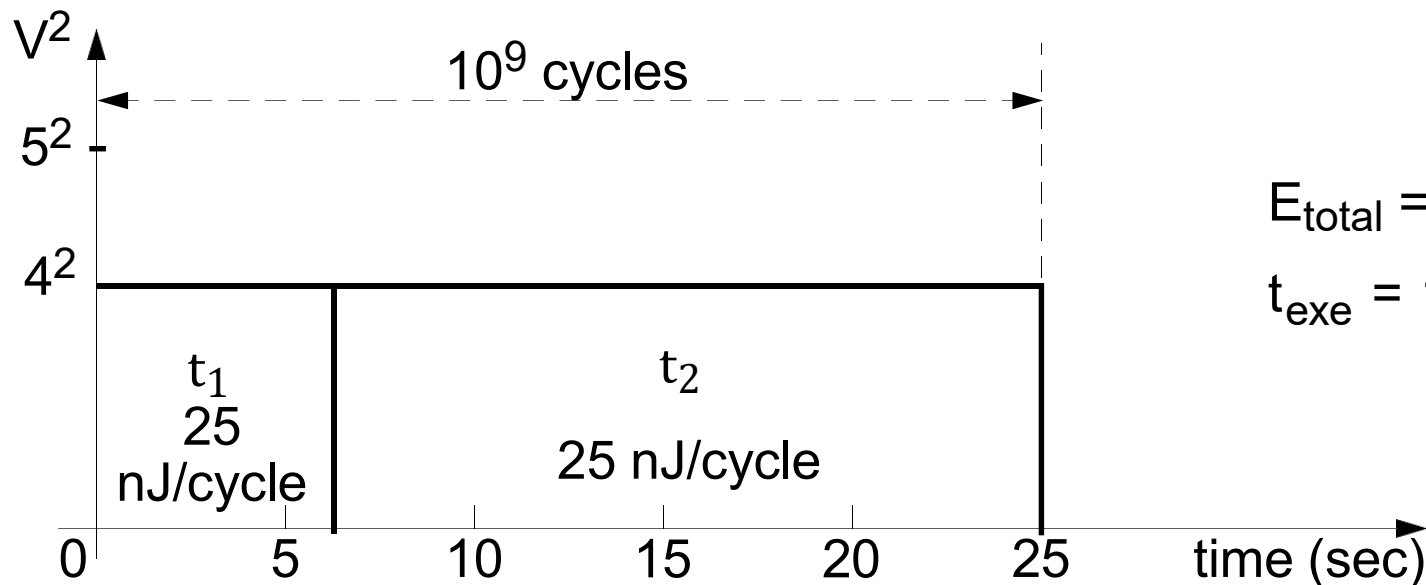
The Basic Principle

- We consider two tasks t_1 and t_2 :
 - Computation t_1 : 250×10^6 execution cycles; t_2 : 750×10^6 execution cycles
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - energy: 40 nJ/cycle at nominal voltage; at 4V: $40 \times 4^2 / 5^2 = 25 \text{ nJ/cycle}$
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage;
at 4V: $50 \times 4 / 5 = 40 \text{ MHz}$ (40×10^6 cycles/sec).



The Basic Principle

- We consider two tasks t_1 and t_2 :
 - Computation t_1 : 250×10^6 execution cycles; t_2 : 750×10^6 execution cycles
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - energy: 40 nJ/cycle at nominal voltage; **at 4V: $40 \times 4^2 / 5^2 = 25 \text{ nJ/cycle}$**
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage;
at 4V: $50 \times 4 / 5 = 40 \text{ MHz}$ (40×10^6 cycles/sec).



$$E_{\text{total}} = 10^9 \times (25 \times 10^{-9}) = 25 \text{ J}$$

$$t_{\text{exe}} = 10^9 / (40 \times 10^6) = 25 \text{ sec}$$

Considering Task Particularities

- Energy consumed by a task:

$$E = \frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}$$

N_{SW} = number of gate transitions per clock cycle.

C = switched capacitance per clock cycle.

- Average energy consumed by task per cycle:

$$E_{CY} = \frac{1}{2} \times C \times V_{DD}^2 \times N_{SW}$$

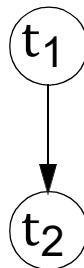
- Often tasks differ from each other in terms of executed operations → N_{SW} and C differ from one task to the other.



The average energy consumed per cycle differs from task to task.

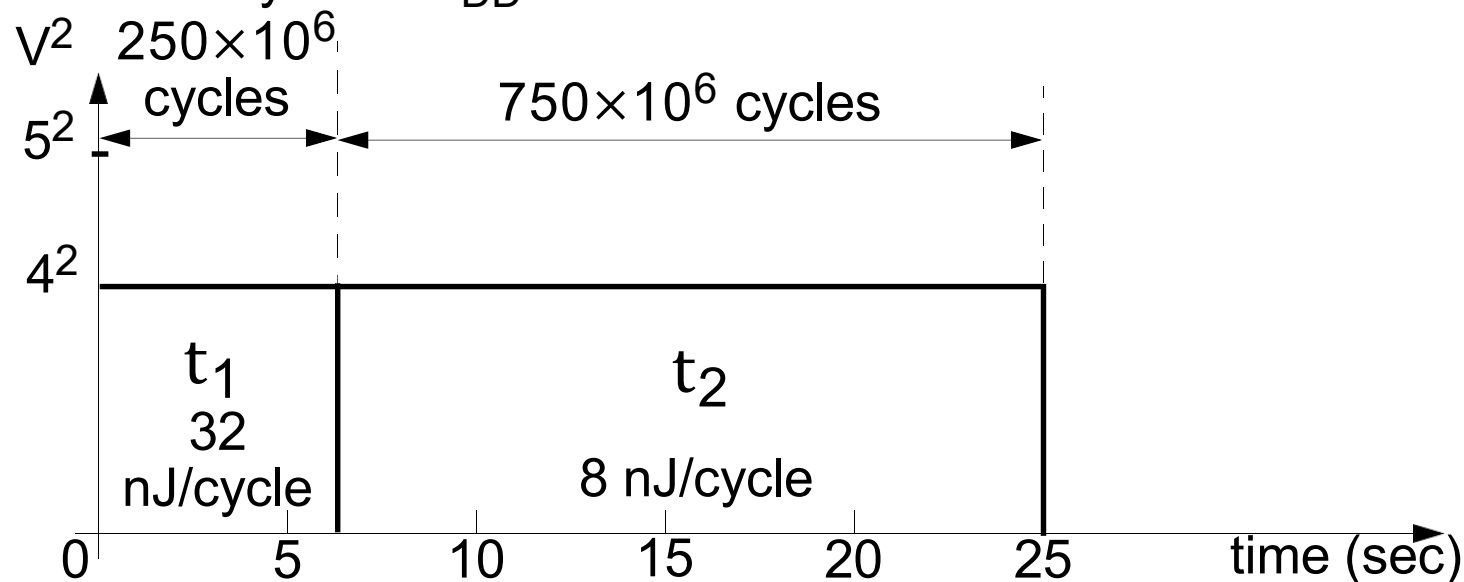
Considering Task Particularities

- We consider two tasks t_1 and t_2 :
 - Computation t_1 : 250×10^6 execution cycles; t_2 : 750×10^6 execution cycles
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage;
at 4V: $50 \times 4/5 = 40$ MHz (40×10^6 cycles/sec).
at 2.5V: $50 \times 2.5/5 = 25$ MHz (25×10^6 cycles/sec).
 - Energy t_1
 - 50 nJ/cycle at $V_{DD} = 5V$.
 - 32 nJ/cycle at $V_{DD} = 4V$.
 - 12.5 nJ/cycle at $V_{DD} = 2.5V$.
 - Energy t_2
 - 12.5 nJ/cycle at $V_{DD} = 5V$.
 - 8 nJ/cycle at $V_{DD} = 4V$.
 - 3 nJ/cycle at $V_{DD} = 2.5V$.



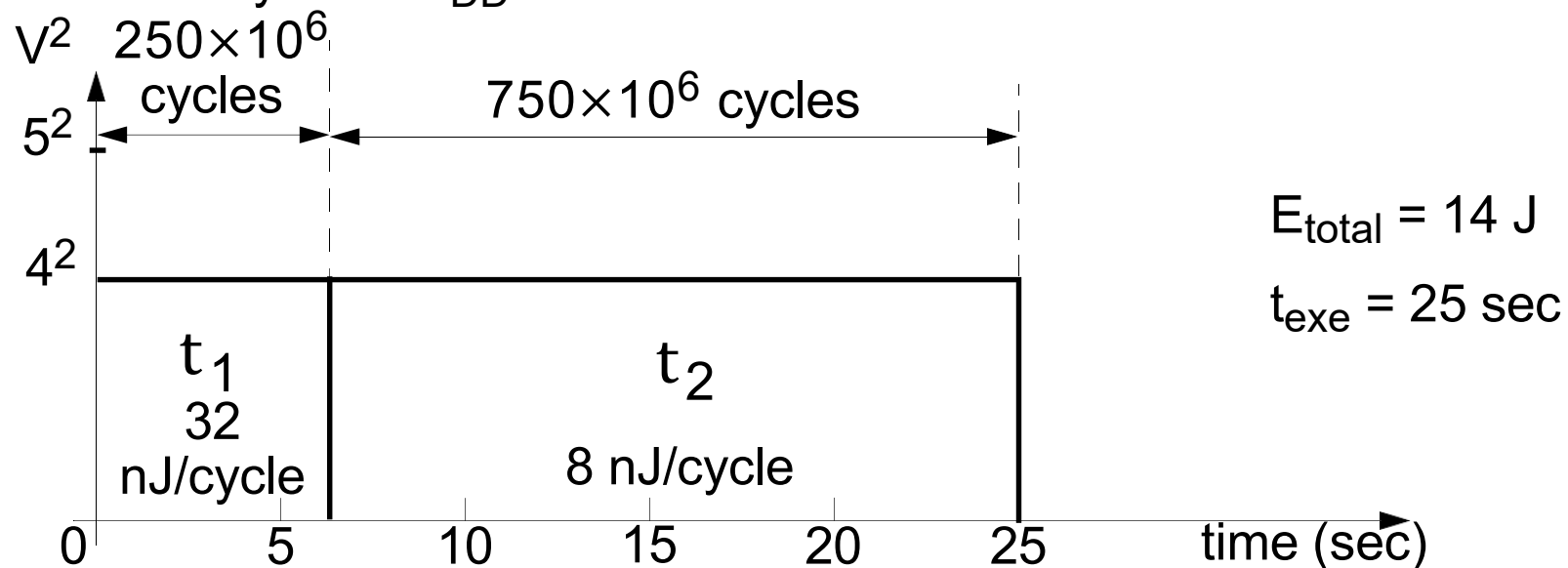
Considering Task Particularities

- We consider two tasks t_1 and t_2 :
 - Computation t_1 : 250×10^6 execution cycles; t_2 : 750×10^6 execution cycles
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage;
at 4V: $50 \times 4/5 = 40$ MHz (40×10^6 cycles/sec).
at 2.5V: $50 \times 2.5/5 = 25$ MHz (25×10^6 cycles/sec).
 - Energy t_1
 - 50 nJ/cycle at $V_{DD} = 5$ V.
 - 32 nJ/cycle at $V_{DD} = 4$ V.
 - 12.5 nJ/cycle at $V_{DD} = 2.5$ V.
 - Energy t_2
 - 12.5 nJ/cycle at $V_{DD} = 5$ V.
 - 8 nJ/cycle at $V_{DD} = 4$ V.
 - 3 nJ/cycle at $V_{DD} = 2.5$ V.



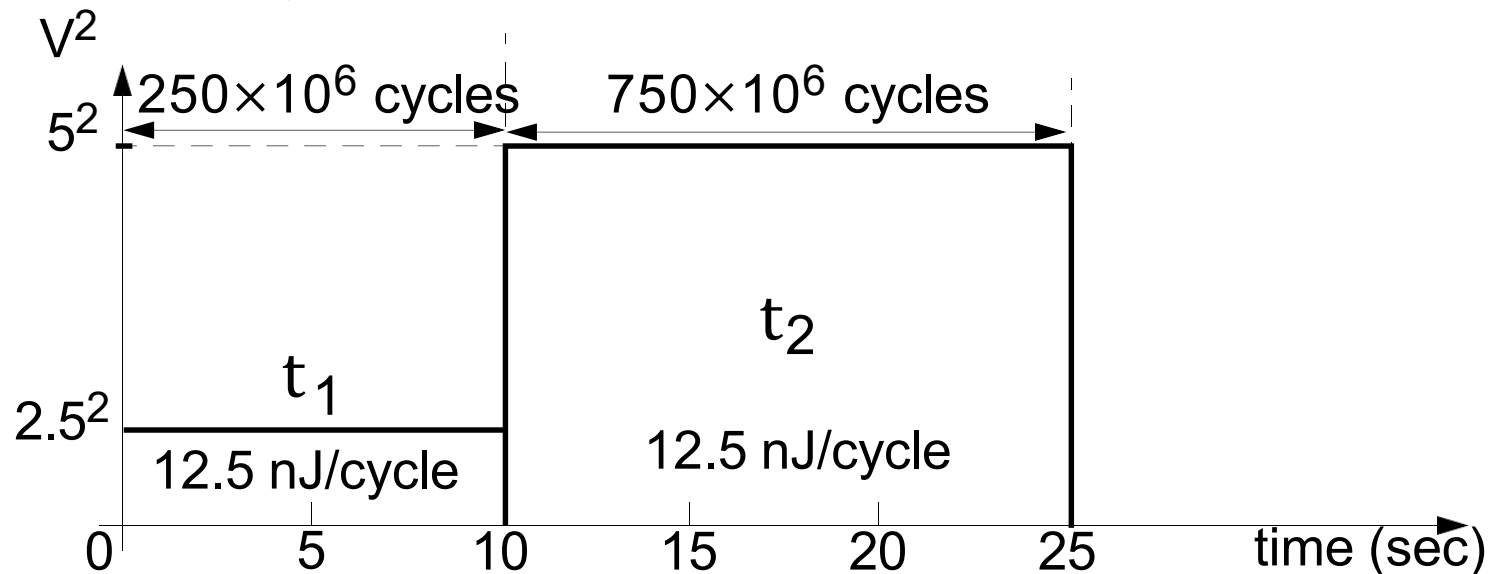
Considering Task Particularities

- We consider two tasks t_1 and t_2 :
 - Computation t_1 : 250×10^6 execution cycles; t_2 : 750×10^6 execution cycles
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage;
 at 4V: $50 \times 4/5 = 40$ MHz (40×10^6 cycles/sec).
 at 2.5V: $50 \times 2.5/5 = 25$ MHz (25×10^6 cycles/sec).
 - Energy t_1
 - 50 nJ/cycle at $V_{DD} = 5$ V.
 - 32 nJ/cycle at $V_{DD} = 4$ V.
 - 12.5 nJ/cycle at $V_{DD} = 2.5$ V.
 - Energy t_2
 - 12.5 nJ/cycle at $V_{DD} = 5$ V.
 - 8 nJ/cycle at $V_{DD} = 4$ V.
 - 3 nJ/cycle at $V_{DD} = 2.5$ V.



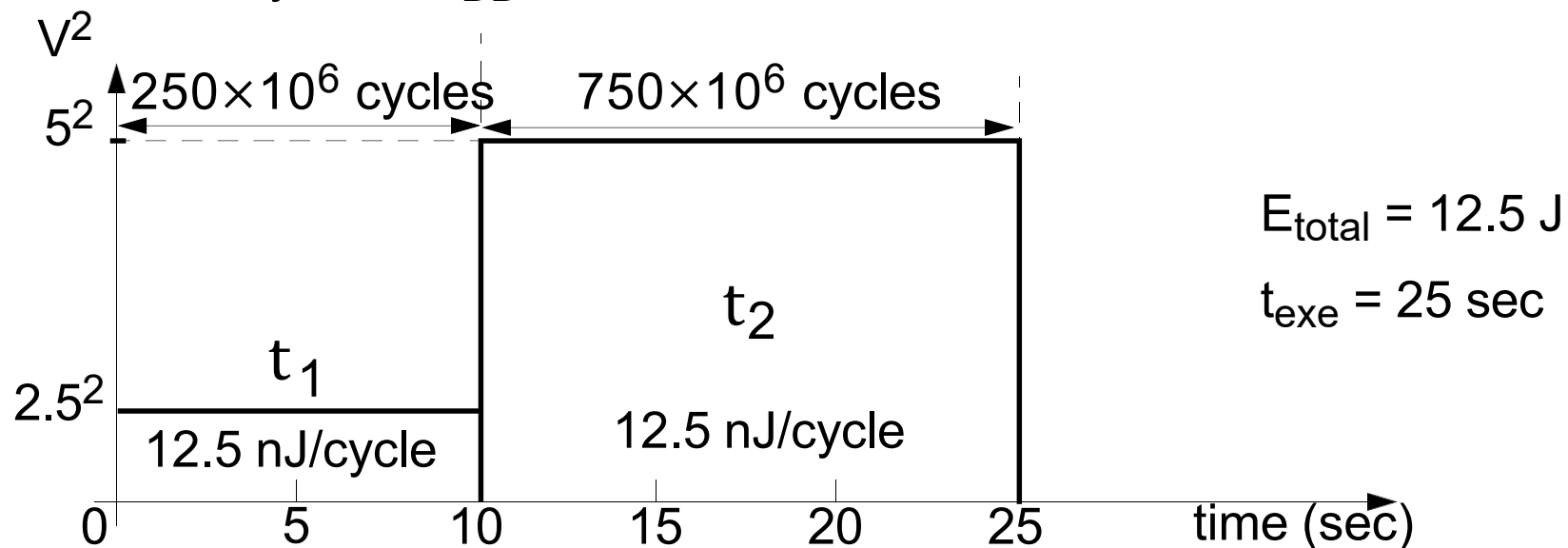
Considering Task Particularities

- We consider two tasks t_1 and t_2 :
 - Computation t_1 : 250×10^6 execution cycles; t_2 : 750×10^6 execution cycles
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage;
at 4V: $50 \times 4/5 = 40$ MHz (40×10^6 cycles/sec).
at 2.5V: $50 \times 2.5/5 = 25$ MHz (25×10^6 cycles/sec).
 - Energy t_1
 - 50 nJ/cycle at $V_{DD} = 5V$.
 - 32 nJ/cycle at $V_{DD} = 4V$.
 - 12.5 nJ/cycle at $V_{DD} = 2.5V$.
 - Energy t_2
 - 12.5 nJ/cycle at $V_{DD} = 5V$.
 - 8 nJ/cycle at $V_{DD} = 4V$.
 - 3 nJ/cycle at $V_{DD} = 2.5V$.



Considering Task Particularities

- We consider two tasks t_1 and t_2 :
 - Computation t_1 : 250×10^6 execution cycles; t_2 : 750×10^6 execution cycles
 - deadline: 25 seconds.
 - processor nominal (maximum) voltage: 5V.
 - processor speed: 50MHz (50×10^6 cycles/sec) at nominal voltage;
 at 4V: $50 \times 4/5 = 40$ MHz (40×10^6 cycles/sec).
 at 2.5V: $50 \times 2.5/5 = 25$ MHz (25×10^6 cycles/sec).
 - Energy t_1
 - 50 nJ/cycle at $V_{DD} = 5V$.
 - 32 nJ/cycle at $V_{DD} = 4V$.
 - 12.5 nJ/cycle at $V_{DD} = 2.5V$.
 - Energy t_2
 - 12.5 nJ/cycle at $V_{DD} = 5V$.
 - 8 nJ/cycle at $V_{DD} = 4V$.
 - 3 nJ/cycle at $V_{DD} = 2.5V$.



Considering Task Particularities

- If power consumption per cycle differs from task to task the “basic principle” is no longer true!

Voltage levels have to be reduced with priority for those tasks which have a larger energy consumption per cycle.

- One individual voltage level has to be established for each task, so that deadlines are just satisfied.

Discrete Voltage Levels

- Practical microprocessors can work only at a finite number of discrete voltage levels.



The “ideal” voltage V_{ideal} , determined for a certain task does not exist.

Discrete Voltage Levels

- Practical microprocessors can work only at a finite number of discrete voltage levels.



The “ideal” voltage V_{ideal} , determined for a certain task does not exist.

- A task is supposed to run for time t_{exe} at the voltage V_{ideal} .
On the particular processor the two closest available neighbours to V_{ideal} are:
 $V_1 < V_{\text{ideal}} < V_2$.



You have minimised the energy if you run the task for time t_1 at voltage V_1 and for t_2 at voltage V_2 , so that $t_1 + t_2 = t_{\text{exe}}$.

The Pitfalls with Ignoring Leakage

$$P = \underbrace{\frac{1}{2} \times C \times V_{DD}^2 \times f \times N_{SW}}_{\text{dynamic}} + \underbrace{Q_{SC} \times V_{DD} \times f \times N_{SW}}_{\text{dynamic}} + \underbrace{I_{leak} \times V_{DD}}_{\text{static}}$$

<u>Switching power</u> Power required to charge/discharge circuit nodes	<u>Short-circ. power</u> Dissipation due to short-circuit current	<u>Leakage power</u> Dissipation due to leakage current
--	--	--

The Pitfalls with Ignoring Leakage

$$E = \underbrace{\frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}}_{\text{dynamic}} + \underbrace{L_g \times (V_{dd} \times K_3 \times e^{K_4 \times V_{dd}} \times e^{K_5 \times V_{bs}} + |V_{bs}| \times I_{ju}) \times t}_{\text{leakage}}$$

C = node capacitances

N_{SW} = switching activities
(number of gate transitions
per clock cycle)

N_{CY} = number of cycles needed
for the task.

f = frequency of operation

V_{DD} = supply voltage

K_{3..5} = technology dependent constants

L_g = number of gates

V_{bs} = body-bias voltage

I_{ju} = body junction leakage current

The Pitfalls with Ignoring Leakage

$$E = \underbrace{\frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}}_{\text{dynamic}} + \underbrace{L_g \times (V_{dd} \times K_3 \times e^{K_4 \times V_{dd}} \times e^{K_5 \times V_{bs}} + |V_{bs}| \times I_{ju}) \times t}_{\text{leakage}}$$

Minimise this and
ignore the rest!

The Pitfalls with Ignoring Leakage

$$E = \underbrace{\frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}}_{\text{dynamic}} + \underbrace{L_g \times (V_{dd} \times K_3 \times e^{K_4 \times V_{dd}} \times e^{K_5 \times V_{bs}} + |V_{bs}| \times I_{ju}) \times t}_{\text{leakage}}$$

Minimise this and
ignore the rest!



1. We don't optimize global energy but only a part of it!
2. We can get it even very wrong and increase energy consumption!

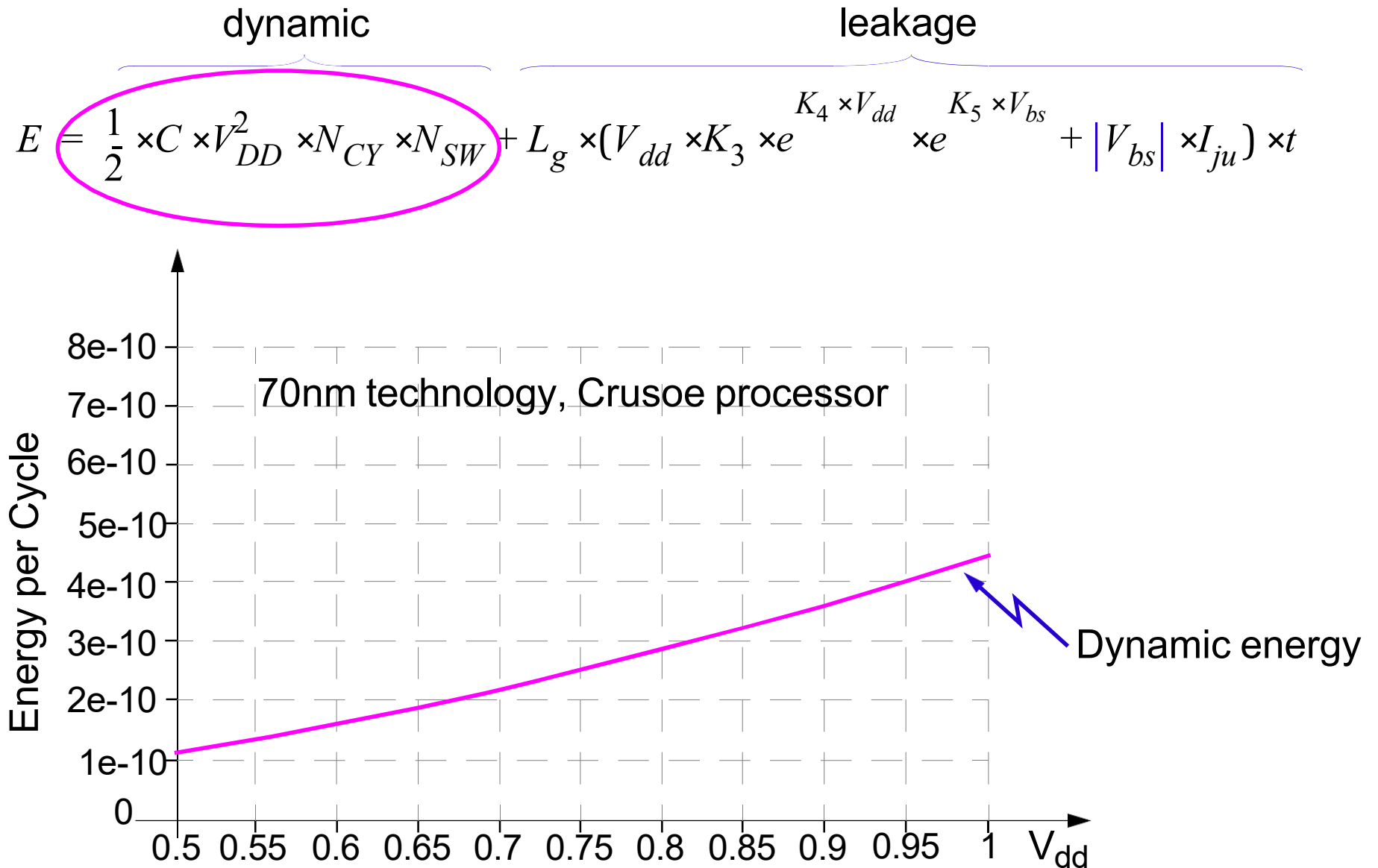
The Pitfalls with Ignoring Leakage

$$E = \underbrace{\frac{1}{2} \times C \times V_{DD}^2 \times N_{CY} \times N_{SW}}_{\text{dynamic}} + \underbrace{L_g \times (V_{dd} \times K_3 \times e^{K_4 \times V_{dd}} \times e^{K_5 \times V_{bs}} + |V_{bs}| \times I_{ju}) \times t}_{\text{leakage}}$$

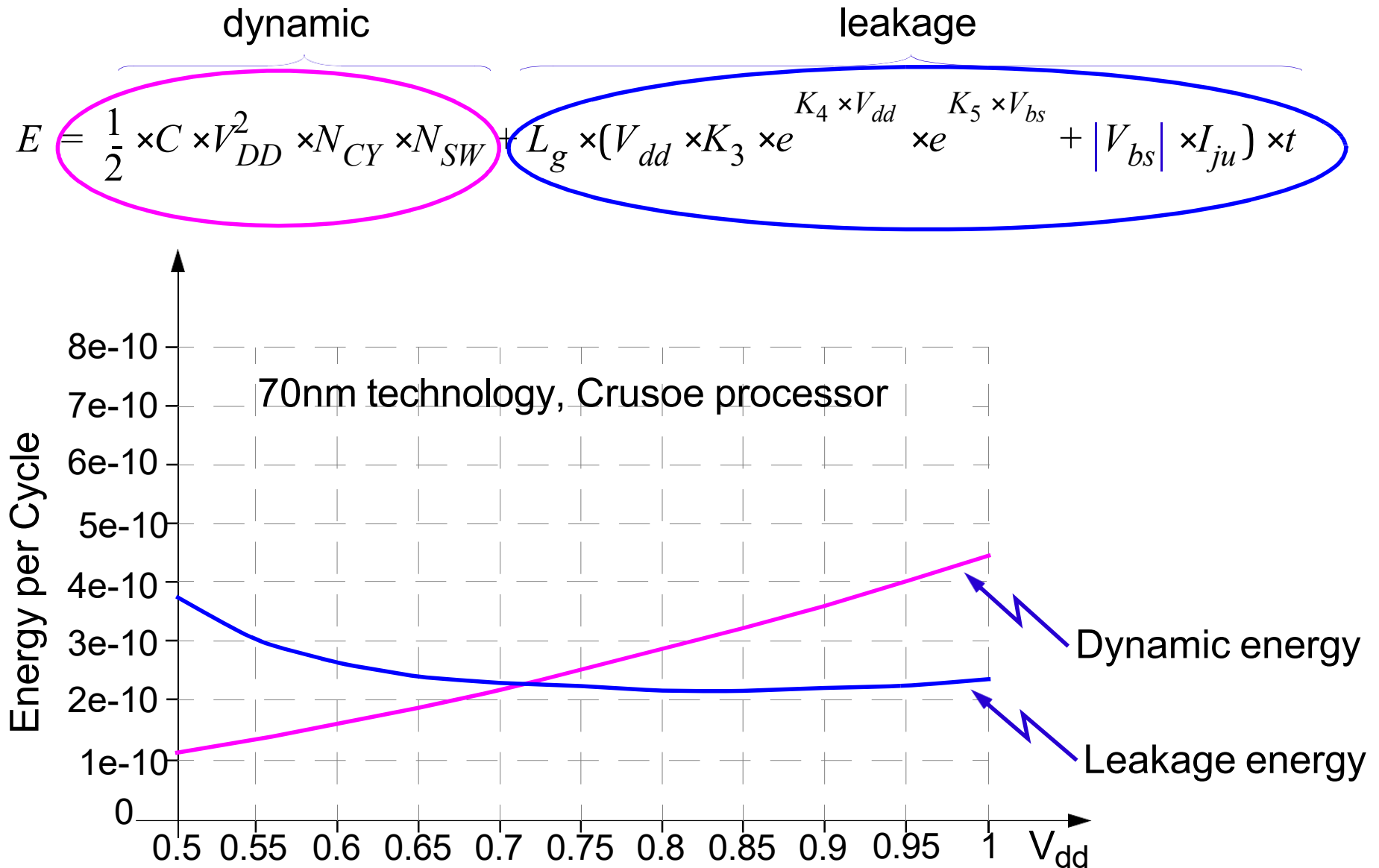
This one decreases with V_{dd} regardless of increased time.

This one decreases with V_{dd} , but grows with time!

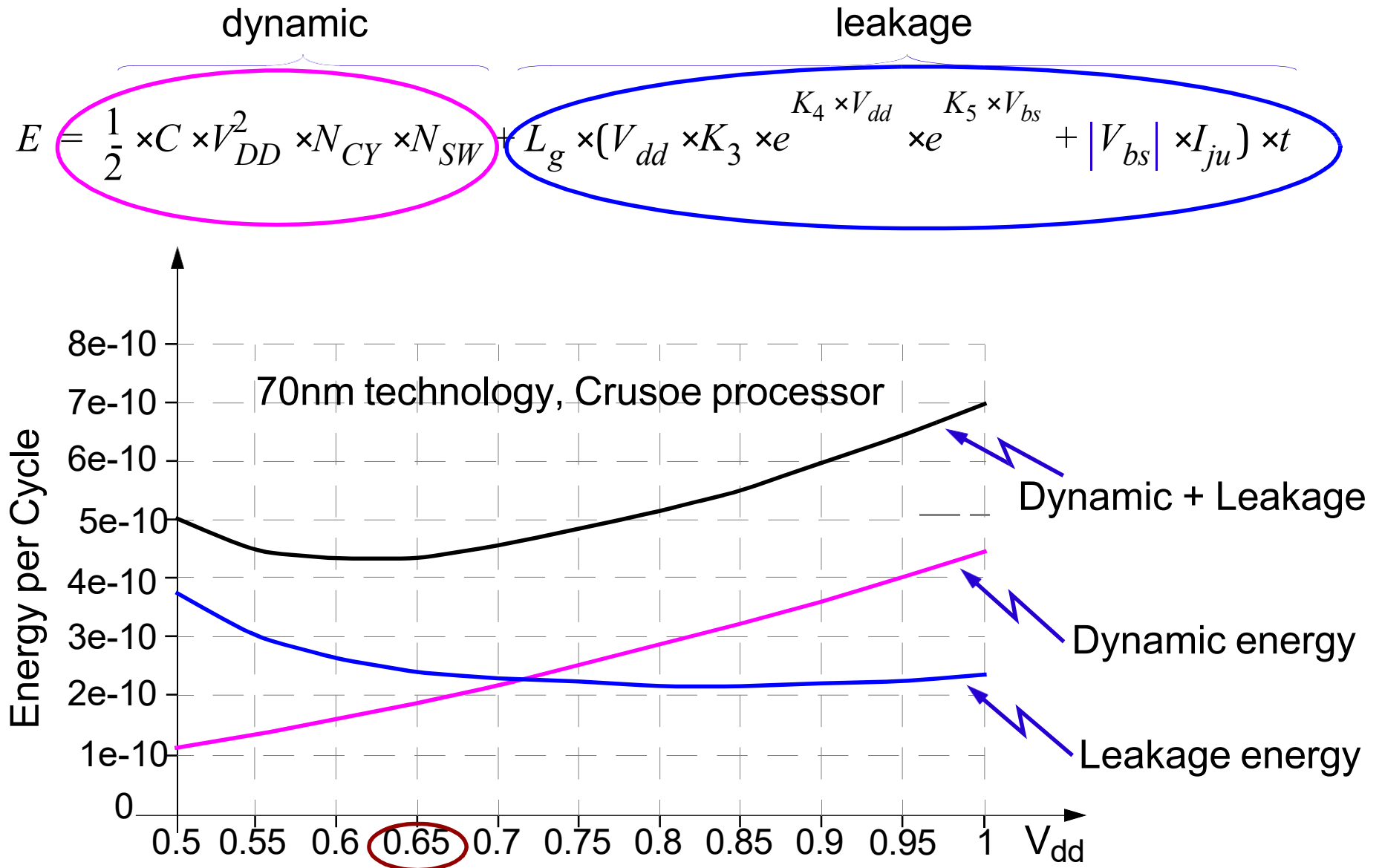
The Pitfalls with Ignoring Leakage



The Pitfalls with Ignoring Leakage



The Pitfalls with Ignoring Leakage



The Pitfalls with Ignoring Leakage

